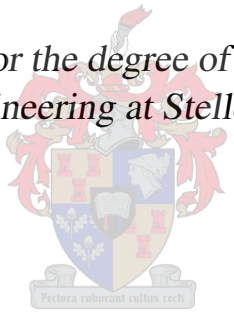
A Vision-based South African Sign Language Tutor

by

Hendrik Adrianus Cornelis de Villiers

Dissertation presented for the degree of Doctor of Philosophy in the Faculty of Engineering at Stellenbosch University



Promoters:

Prof. T. R. Niesler Prof. L. van Zijl

April 2014

Declaration

By submitting this dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: April 2014 . . . . . .

Abstract

A Vision-based South African Sign Language Tutor

H. A. C. de Villiers

Dissertation: PhD

April 2014

A sign language tutoring system capable of generating detailed context-sensitive feedback to the user is presented in this dissertation. This stands in contrast with existing sign language tutor systems, which lack the capability of providing such feedback.

A domain specific language is used to describe the constraints placed on the user's movements during the course of a sign, allowing complex constraints to be built through the combination of simpler constraints. This same linguistic description is then used to evaluate the user's movements, and to generate corrective natural language feedback. The feedback is dynamically tailored to the user's attempt, and automatically targets that correction which would require the least effort on the part of the user. Furthermore, a procedure is introduced which allows feedback to take the form of a simple to-do list, despite the potential complexity of the logical constraints describing the sign. The system is demonstrated using real video sequences of South African Sign Language signs, exploring the different kinds of advice the system can produce, as well as the accuracy of the comments produced.

To provide input for the tutor system, the user wears a pair of coloured gloves, and a video of their attempt is recorded. A vision-based hand pose estimation system is proposed which uses the Earth Mover's Distance to obtain hand pose estimates from images of the user's hands. A two-tier search strategy is employed, first obtaining nearest neighbours using a simple, but related, metric. It is demonstrated that the two-tier system's accuracy approaches that of a global search using only the Earth Mover's Distance, yet requires only a fraction of the time. The system is shown to outperform a closely related system on a set of 500 real images of gloved hands.

Uittreksel

'n Visie-gebaseerde Suid-Afrikaanse Gebaretaaltutor

("A Vision-based South African Sign Language Tutor")

H. A. C. de Villiers

Proefskrif: PhD

April 2014

'n Gebaretaaltutorstelsel met die vermoë om konteks-sensitiewe terugvoer te lewer aan die gebruiker word uiteengesit in hierdie proefskrif. Hierdie staan in kontras met bestaande tutorstelsels, wat nie hierdie kan bied vir die gebruiker nie.

'n Domein-spesifieke taal word gebruik om beperkinge te definieer op die gebruiker se bewegings deur die loop van 'n gebaar. Komplekse beperkinge kan opgebou word uit eenvoudiger beperkinge. Dieselfde linguistieke beskrywing van die gebaar word gebruik om die gebruiker se bewegings te evalueer, en om korrektiewe terugvoer te genereer in teksvorm. Die terugvoer word dinamies aangepas met betrekking tot die gebruiker se probeerslag, en bepaal outomaties die maklikste manier wat die gebruiker sy/haar fout kan korrigeer. 'n Prosedure word uiteengesit om die terugvoer in 'n eenvoudige lysvorm aan te bied, ongeag die kompleksiteit van die linguistieke beskrywing van die gebaar. Die stelsel word gedemonstreer aan die hand van opnames van gebare uit Suid-Afrikaanse Gebaretaal. Die verskeie tipes terugvoer wat die stelsel kan lewer, asook die akkuraatheid van hierdie terugvoer, word ondersoek.

Om vir die tutorstelsel intree te bied, dra die gebruiker 'n stel gekleurde handskoene. 'n Visie-gebaseerde handvormafskattingstelsel wat gebruik maak van die Aardverskuiwersafstand (Earth Mover's Distance) word voorgestel. 'n Twee-vlak soekstrategie word gebruik. 'n Rowwe afstandsmate word gebruik om 'n stel voorlopige handpostuurkandidate te verkry, waarna die stel verfyn word deur gebruik van die Aardverskuiwersafstand. Dit word gewys dat hierdie benaderde strategie se akkuraatheid grens aan die van eksakte soektogte, maar neem slegs 'n fraksie van die tyd. Toetsing op 'n stel van 500 reële beelde, wys dat hierdie stelsel beter presteer as 'n naverwante stelsel uit die literatuur.

Acknowledgements

I would like to express my sincere gratitude to the following people and organisations:

- The Wilhelm Frank Trust, Telkom SA and the National Research Foundation of South Africa (grant UID 71926) for their financial support of this research.
- The High Performance Computing facility at Stellenbosch University, which was utilised to perform the experimental evaluation of the hand pose estimation system.
- SLED (`http://www.sled.org.za`) for the sign language classes they offer, which provided valuable insight into the performance requirements of the hand pose estimation system, and whose educational materials served as an invaluable resource for the sign definitions used in the proof of concept system (Note: SLED is not associated with our group and were not involved in the research process)
- My test subjects, for long hours spent in front of the camera.
- My supervisors Prof. Thomas Niesler and Prof. Lynette van Zijl, thank you for your unswerving support, encouragement and guidance.
- Viola Lengner, who still keeps me on track.
- My family, for all your support.
- All of my friends, for your excellent company and keeping me in high spirits.
- My partner Halford, for endless enthusiasm, love and support, without which this would scarcely have been possible.

Dedication

For Halford

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

EMD	Earth Mover's Distance
HMM	Hidden Markov Model
SASL	South African Sign Language

Features

$\bar{x}$	Input features
$\tilde{x}$	Feature constraint variables
$\hat{x}$	Suggested features

Constraints

c_n	Constraint
$s_n(x)$	Satisfaction function
$c_n \leftrightarrow s_n(x)$	The constraint c_n has satisfaction function $s_n(x)$

Metrics

d	A dissimilarity measure or the compound EMD
d^*	Areas-and-means distance measure
d_{SC}	Symmetric chamfer distance
d_{AC}	Asymmetric chamfer distance
d_{EMD}	Earth Mover's Distance

Chapter 1

Introduction

Existing sign language tutors lack the ability to generate context-sensitive advice based on the many complex constraints a signer must satisfy during the phases of a sign. In this work, a context-sensitive advice generation system for a South African Sign Language (SASL) tutor is presented.

Typically, sign language tutors show the user the correct signing of a word, and the user is then asked to attempt the sign. The tutor must then monitor the user's movements, comment on the correctness of the signing and provide advice to the user. It is in particularly this last phase where existing tutors are lacking. It is here that context-sensitive advice would be most useful. Such advice should:

- be specifically tailored to the user's current attempt;
- take into account the easiest way that the user can correct their mistake, given their current attempt;
- take into account the exact constraints on the user's movements during any particular phase of a sign;

In addition, it is desirable that the advice expresses the available information in a form that the user can easily understand and is not self-contradictory. Furthermore, the system must allow easy extension by allowing the addition of new constraints and new modes of providing feedback such as natural language feedback, diagrams or animations.

It is the current lack of a satisfactory solution to this problem that this dissertation seeks to address, and it will be demonstrated that these requirements can be met simultaneously.

The system is vision-based, as a computing device with a video camera is possibly the lowest common denominator that can serve as a target for development. This necessitates a subsystem which can accurately track the relevant portions of the user's movements. While the user's hands are by no means the only important aspect during signing, hand pose estimation is one of the most challenging subdisciplines of human pose estimation, due to the variability in hand shape and orientation in sign languages. Movements of interest are rapid, yet subtle.

These complexities are compounded by the requirements imposed by a tutor system. While it is possible to perform sign language *recognition* using features that do not include any detailed reconstruction of the hand pose, a tutor system is tasked with providing advice given a sign that is known *a priori*. As such, the problem is one of *producing commentary and advice*, and not one of recognition. Therefore, a more detailed reconstruction of the hand pose must be attempted. As there is also an expectation that the sign will be performed incorrectly (at least initially), the system must leave open the possibility of hand positions, orientations and shapes that are far from correct. This implies that an estimation of the current hand pose is needed that cannot rely on prior pose information.

There exists a class of hand pose estimation systems that make use of nearest neighbour search to find a set of candidate poses without using prior information. However, the similarity measure defining "nearness" varies greatly between systems. A similarity metric that has shown considerable promise in recent years is the Earth Mover's Distance (EMD). The EMD is, however, relatively expensive to calculate, and so any practical system employing it must use it sparingly. The hand pose estimation system presented in Part I utilises the EMD, but addresses the concerns surrounding its computational complexity by using it only in the last stage of the hand pose estimation process.

The key contributions stemming from the work presented in this dissertation can now be listed. The following are addressed in Part I, which describes the hand pose estimation component of the system.

- A system is presented which avoids excessive reliance on the EMD by means of a two-tier search. First, initial searches for candidates are performed with a related but easily computed approximate metric. Then, the EMD is used to refine this initial set of candidates.

- The system is tested on a set of 500 real images of gloved hands. It is confirmed that the two-tier system's performance approaches that of a system which directly uses the EMD for all distance calculations, while offering a large reduction in computational complexity. The accuracy of the system is compared to a closely related existing system, and found to improve upon it.

- An automatic colour calibration procedure utilising a graphical model is proposed, and it is demonstrated that this procedure is able to provide accurate colour model estimates based on rough prior knowledge of the colour models, and a calibration image within which the user assumes a known hand pose.

Part II of the dissertation describes the tutor component of the system, and addresses the following.

- A domain-specific language is defined which allows the description of sign language signs by defining the constraints on the user's movement during each phase of a sign. The language allows complex constraints to be built from atomic constraints using a variety

of logical connectives such as ‘and’, ‘or’ and ‘not’. The language can be easily extended by defining new atomic constraints.

- An advice generation system is presented that uses the linguistic description of the sign to generate context-sensitive feedback aimed at correcting the user’s attempt. The system takes into account the relevant aspects of the user’s current attempt, minimises the effort needed on the part of the user to correct their mistake, and targets a set of suggested features which satisfies all the constraints on the user’s movement.

- A procedure is presented which allows the potentially complex logical structure of the constraints placed on the user’s movement to be simplified into a simple to-do list of advice which the user may more readily understand, yet does not discard relevant information. This procedure also eliminates potential sources of paradoxical advice, and ensures that constraints may generate advice independent of each other.

- The advice generation system leaves open the exact mode of feedback given to the user. Possible modes of feedback include natural language feedback, diagrams or animations (natural language feedback is employed for purposes of demonstration).

- The system is evaluated using real video sequences obtained for six South African Sign Language signs. The system behaviour is explored for a variety of constraint types, and results are presented regarding the types and quality of the advice produced by the system.

Publications stemming from this work include De Villiers et al. [26] (under review), which presents the tutor system, and De Villiers et al. [24] (published), which discusses the hand pose estimation system.

The remainder of the dissertation will be presented in three parts. The first part presents the hand pose estimation system, the second describes the tutor component of the system, and the final part draws conclusions from the results obtained in the previous two parts, and explores potential avenues for future research.

Part I

Hand Pose Estimation

Chapter 2

Introduction to Hand Pose Estimation

Hand position, velocity, orientation and shape are core elements used to convey meaning in sign languages. Information about the state of the hands is thus key to any natural language processing system for signed languages.

Certain systems collect hand information by using data gloves [58]. In these cases, extracting information about the hand pose is especially simple. Additional sensors can be used to obtain information about hand orientation and position (and thus velocity). While attractive in terms of the state information they can provide, data gloves are expensive, and are a relatively rare commodity.

In addition, no information about non-manual gestures can be obtained using data gloves, and so some other means of gathering such data is needed, further increasing costs.

In contrast, vision-based hand pose estimation systems make use of camera hardware, which is now ubiquitous, and the price of quality hardware continues to fall as technology advances. Apart from hand pose estimation, non-manual gestures may also be tracked using video hardware, and so no extra investment in equipment is necessary.

However, hand pose estimation from video sequences is non-trivial. While tracking the position and the velocity of the hand are relatively easy, estimating the orientation and shape of the hand is more challenging. There are several reasons for the complexity of hand pose estimation. The hand has a large number of degrees of freedom, and so the state space that needs to be explored is large. In addition, this makes it challenging to collect a significant quantity of training data for machine learning approaches to hand pose estimation.

When the task of interest is sign language recognition, it is often not necessary to attempt an accurate reconstruction of the hand state. Features which encapsulate the appearance of the hand, rather than its pose, may easily be rich enough to choose between the discrete possibilities present in different signs.

Sign language tutors, however, must comment on the shape of the hand, and provide advice regarding it when necessary. This implies that some detailed reconstruction of the hand pose must be attempted.

Yet, at the same time, processing sign language represents a particularly challenging subdo-

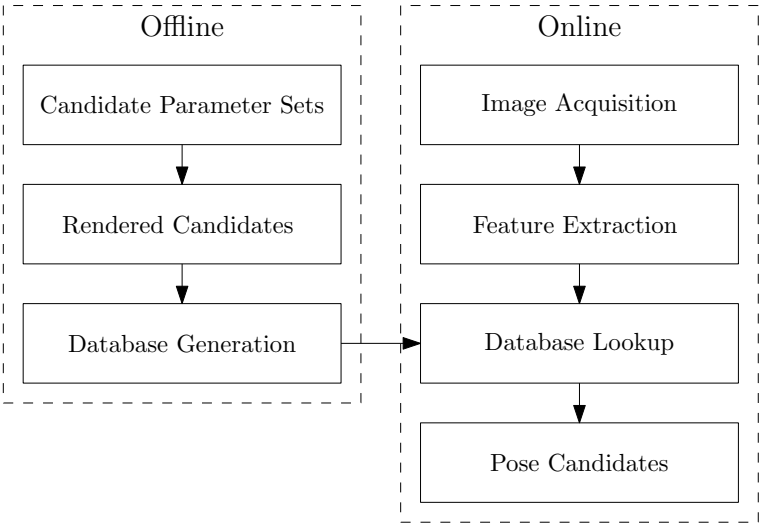


Figure 2.1: Pose estimation system structure, indicating offline and online components.

main of gesture recognition. The movements that sign language signs consist of are rapid, yet subtle, placing strong requirements on such systems.

2.1 Pose estimation through database search

With the growth in availability of computer memory, searching large databases for pose candidates has become a viable option for obtaining pose estimates. Database driven pose estimation systems are composed of several components, which can be divided into an offline and an online part. A conceptual overview is shown in Figure 2.1.

The offline part of such systems is responsible for creating each entry in the database. While it is technically possible to populate the database with real images, the sheer number of possible pose parameters makes this impractical. Typically the offline part renders synthetic images based on a set of pose parameters for each entry, and extracts features from the images. The feature extraction procedure applied to the synthetic database images is effectively the same procedure that real input images undergo, and so they can be compared for similarity.

An index structure, which can be searched at runtime for matching features, is created. During index creation, each possible pose candidate is considered and compared with others, and as such is inherently slower than the online system. However, since the offline preprocessing is performed only once, speed is not a primary consideration. The tradeoff between the storage needed for pregenerated information and synthesising it at runtime is explored in De Villiers et al. [25].

The online part of the system processes input images to locate and track the hand, performs feature extraction and searches the index structure constructed during the offline phase to obtain the closest matches of the input image. The pose parameters of synthetic images that are similar to the input images are returned as potential pose candidates.

This type of system simultaneously addresses several of the concerns already mentioned. Firstly, as users of sign language tutors are expected to make errors, at least initially, tutor systems may not rely on prior information regarding hand pose, meaning that the entire state space of the hand needs to be considered in some way. Because the search takes the entire database into consideration, such systems often need no initialisation. This makes them suitable either for standalone frame-by-frame estimation, or as an initialisation step for trackers which do depend on some preliminary pose information.

Secondly, by populating the database using synthetically generated data, domain knowledge of the object to be recognised can be easily incorporated into the estimation system. This addresses the almost inevitable scarcity of representative data, given the large state space associated with the human hand.

A key consideration in the design of database driven pose estimation systems is the measure of similarity used to compare features. Choosing a metric that returns pose candidates that reflect closeness in the underlying pose parameters of the input and the database elements is of primary importance.

In order to ensure that database query resolution occurs at acceptable speeds, some way of limiting the number of necessary comparisons with database elements needs to be provided. The embedding of pose candidates in a metric space [89] represents a simple way of limiting the number of comparisons needed. A metric space is defined by a set of elements and a measure (called a metric) of dissimilarity between those elements. Because distances between elements within a metric space obey the triangle equality, large parts of a database may be eliminated by comparison with single pivot objects. Chapter 4 will described similarity search in metric spaces in more detail.

2.2 The Earth Mover's Distance

The Earth Mover's Distance (EMD) is a metric that has shown promise as a means of measuring dissimilarity between point sets [56, 57]. Intuitively, it is the minimum amount of work needed to redistribute the point set representing one image (piles of "earth") to cover the point set of another image (consisting of "holes"). A more formal discussion of the EMD is provided in Section 4.2.

The EMD is, however, a computationally expensive operation. In systems which employ it, economising on its use can lead to notable speed gains. At the same time, such schemes should not excessively degrade the system accuracy.

2.3 Overview

In this part of the dissertation, an approximation to exact query resolution using the EMD is proposed in the context of hand pose estimation.

A coloured glove is used to emphasise the different fingers so that greater fidelity in pose reconstruction may be achieved. This also imposes a measure of uniformity on the input data, which allows the system to cater for a greater variety of users. Because the pose estimation system is to be used as a frontend for the tutor system, the use of a coloured glove is justified, as e-learning environments are more amenable to control than general settings. Note that the work detailed here was published in De Villiers et al. [24]. The presentation will follow this article, with extensions introduced after its publication being described in Chapters 7 and 8.

A two-tier search approach is demonstrated that uses an initial set of candidates obtained using a rough, but easily evaluated, metric. The set is then refined by applying the EMD. It is demonstrated quantitatively that approximate queries retain much of the accuracy of exact queries using the EMD directly. Furthermore, it is shown that the system outperforms a closely related hand pose estimation system, based on an evaluation using 500 real images of gloved hands with known ground truth.

In Chapter 3, an overview of hand pose estimation is provided. The properties of different pose estimation approaches are discussed. Finally, pose estimation systems closely related to the one presented in this dissertation are described and their characteristics explored. This provides the context to place the pose estimation system within the family of systems to which it belongs.

The theoretical foundations of similarity search in metric spaces are reviewed in Chapter 4, followed by a discussion of the EMD.

The hand pose estimation system itself is described in Chapter 5, while detailing each aspect of its operations with reference to similarities with and differences from existing systems.

In Chapter 6, the system is evaluated. It is demonstrated that approximate queries approach the accuracy of exact queries using the EMD while providing reductions in the time necessary to process a query. In addition, the system accuracy is compared with an existing system using real images obtained of a gloved hand.

An automatic colour calibration system is presented Chapter 7 which employs a graphical model to infer colour models given a calibration image.

Extensions to the pose estimation system necessary to track hand poses over video sequences are related in Chapter 8.

Finally, conclusions drawn from the experimental evaluation of the system, and its extensions, are presented in Chapter 9.

Chapter 3

Survey of Hand Pose Estimation Methods

In this chapter, an overview of relevant hand pose estimation literature will be given. The first section will give a broad overview of the field. The second section will discuss techniques for estimating hand pose from single frames of video, with database driven pose estimation systems being such a technique. The remaining sections will consider in detail database driven systems which are closely related to the pose estimation system presented in this dissertation. In the final section, a critique of existing systems will be given which motivates the course of the research presented in subsequent chapters.

3.1 Introduction

Hand pose estimation is a challenging subfield of human body pose estimation, due to a number of factors, the most important being the highly variable nature of the poses a human hand can assume. Detailed models of human hands may have more than twenty degrees of freedom (DOFs). For example, the system presented in this dissertation has four degrees of freedom per finger, three for hand orientation and three for hand position, giving 26 DOFs in total. Self-occlusion is common in hand poses, hiding appearance information needed for providing good estimates of these many degrees of freedom.

Hand pose estimation systems fall into three broad classes [29]. The first class is composed of those systems which attempt only a partial estimation of hand pose parameters. An example can be found in Von Hardenberg and Bérard [78], where the outstretched fingertips of a hand are searched for, and used to establish points of interaction between the user and a user interface projected onto a wall.

Partial pose estimation is not sufficient for sign language tutor software, simply because they must account for all possible variability in the user's signing. So, the second and third group of hand pose estimation systems, systems which attempt a full reconstruction of hand pose, is more relevant to this work. These last two groups include model-based trackers, and single frame pose estimation systems.

Model-based trackers are top-down systems which use dynamic models of the human hand

to establish a frame-by-frame tracking of the hand. Model-based trackers, in general, explore a much smaller state space than single frame pose estimation systems, as they use the last frame of video as a point of departure when estimating the next frame's pose. This can lessen the computational burden on such systems.

Model-based trackers have two chief weaknesses. The first is the need for initialisation. In general, such systems require the user to assume a known pose before the tracking algorithm can lock onto the hand, or they must perform single frame pose estimation at the start of the video sequence. The second weakness of such systems is the possibility of losing the track, meaning the system must be reinitialised. Model-based trackers do employ a variety of different strategies such as multiple hypothesis tracking to avoid losing track of the hand. However, especially in cases where the hands move rapidly, information about the previous hand pose is not necessarily as informative as in more limited settings, negating much of the advantage these trackers gain from taking into account neighbouring frames of video.

Single frame pose estimation systems have emerged recently as a means of solving general hand pose estimation problems [29]. These systems use only the information present in a single frame of video to generate pose candidates. This is inherently a more challenging problem than model-based tracking where a good initial pose estimate is available, and single frame pose estimation systems are in general more computationally complex than model-based alternatives. However, much is gained when a system can successfully estimate poses from a single frame of video. Such systems need no initialisation, although initial calibration of the camera and of any colour models (for example) might be necessary when the system is started for the first time. Without the need for initialisation, single frame pose estimation systems are ideal for (re)initialising model-based trackers, or for use on their own as hand trackers, possibly allowing multiple hypotheses to be tracked over time after single frame estimation has been performed.

The hand pose estimation system presented in this dissertation falls into this last category (single frame pose estimation systems). Because the system is to be used in a sign language tutor, the input is expected to contain rapid and unpredictable changes in hand pose. This is an ideal application for single frame estimation methods. In the next section, these methods will be discussed in more detail.

3.2 Single frame pose estimation systems

The category of single frame hand pose estimation systems, without further qualification, encompasses many types of systems. However, the systems of interest to this discussion are those which are relatively flexible in the poses that they can recognise, and in some way consider the entire state-space either during learning or at runtime. A few systems that exemplify this class of approaches will now be given.

One means of estimating poses from single frames of video is to learn mappings between input features and hand pose parameters. For example, Rosales et al. [55] describe such a

system, where a set of mapping functions ϕ_k are learnt which specialise in different regions of the feature space. Each set of input features is mapped by the mapping functions ϕ_k, producing a candidate pose for each mapping function. The decision about which of the mappings to use as the final set of pose parameters is made using a feedback-matching function, which projects the mapping back into the feature-space. Each backprojection is then compared with the input using a distance measure (Rosales et al. employ the Mahalanobis distance based on the covariance matrix obtained using the training data). The candidate pose with backprojection closest to the input features is selected as the final pose.

It is also possible to use a branched cascade of classifiers to detect hand poses from a large set of possible poses. In such as scheme, a set of rectangular regions of various sizes is considered which covers the entire image. Each of these regions undergo a series of tests. Each of the tests is performed by a classifier which either classifies the region as not containing an object of interest (and so the region is rejected), or it sends the region on for further tests. The Viola-Jones facial detector [72] operates in this manner, with a linear list of such rejection classifiers. A region is marked as containing a face candidate if all the tests succeed in sequence. Stenger et al. [66] use a tree of rejection classifiers to detect an unadorned hand with a fixed shape, but with variable orientation and position. At each non-leaf node of the tree, a region is either rejected, or it is sent to each child node for further testing. Each leaf represents a particular pose, and so if a leaf is reached, that pose has been detected. Stenger et al. demonstrate the system by quantitative testing of a variety of classifiers based on edge or silhouette features, and by showing a number of detection examples.

Another approach is to utilise inverse kinematics to infer the pose of the entire hand, given the location of the fingertips. Nolker and Ritter [50] proposed the GREFIT system, which uses two stages to extract pose parameters from hand images. The hands have a fixed orientation and a highly constrained position. The fingers of the hands are allowed to flex freely. An initial stage detects the positions of the fingertips within the image. Following fingertip detection, parametrised self-organising maps (PSOMs) [79] are used to map the fingertip locations to hand pose parameters. A PSOM is trained for each finger by using a hand model to generate fingertip positions given a set of joint angles (forward kinematics). At runtime, each PSOM then performs the inverse kinematic problem of finding finger joint angles given the fingertip position. The system is demonstrated by showing a number of successful pose detections from still images.

Finally, searching a large database of preprocessed pose candidates for items matching the input image has shown promise in solving hand pose estimation problems. As such a system is the topic of this part of the dissertation, a discussion of existing systems of this type will now be presented.

3.3 Database driven pose estimation systems

In this section, database driven pose estimation systems particularly relevant to this dissertation are first summarised briefly, and then discussed in more detail based on the characteristics which define such systems. The general functioning of this class of system was discussed in Section 2.1.

Athitsos et al. [6–10] demonstrated various systems which utilise a 3D model of an unadorned hand, rendered from several viewpoints in order to obtain database elements. Edge images, extracted from both real and synthetic input, were used as features. The chamfer distance [8] was used as the underlying dissimilarity measure between edge images. Because the chamfer distance is not a metric, a low-distortion embedding into a normed space is applied, allowing efficient spatial data structures to resolve queries in an approximate manner.

Dick, Zieren and Kraiss [27] introduced a conceptually similar system which was used to track a gloved hand for an interactive sign language tutor (detailed in Zieren's PhD thesis [91]). Glove markers included coloured fingers, as well as a rectangular marker on the back of the palm. In order to speed recognition and keep the database compact, simple ellipse-shaped features for each finger were stored and used during comparison, with the Hausdorff distance between ellipses as the underlying metric.

The work of Grauman and Darrell [34] is an example of the Earth Mover's Distance (EMD) being used in some form of pose estimation.[1] Experiments were performed on a database of synthetically rendered silhouettes of human figures in various poses. They apply an embedding of the EMD into L_1, due to Indyk and Thaper [38]. This allows the use of locality-sensitive hashing to speed (approximate) query resolution.

Wang and Popović [80] demonstrated a system which uses a glove with a variety of patches in ten distinguishable colours. As features, they employ small resized versions of the segmented input image, called "tiny images". The distance measure employed, the "tiny image distance", shares certain features with the chamfer distance used by Athitsos et al. [8], as discussed in Section 3.7. To improve search performance, each database element is assigned a 192-bit binary code based on a learnt coding scheme which maps similar images to bit strings such that the Hamming distance between the bit strings is low. Approximate search using the Hamming distance may then be followed by a more detailed treatment using the tiny image distance. After initial pose estimation, inverse kinematics is used to refine the pose estimate.

Schröder et al. [59] recently published an extension of the system described by Wang and Popović which uses depth camera information. However, their explicit aim is extraction of hand location and orientation, rather than detailed hand shape estimation, and so is less relevant to the current discussion.

[1] Note that, while the hand pose estimation system presented by Ren et al. [53] does make use of the EMD, it is fundamentally different from the system presented in this dissertation. Their system currently only processes frontal views of the inside of the hand, and is limited to making decisions about whether or not a finger is raised. Features are not true two-dimensional contour features, but are one-dimensional polar histograms derived from the segmented hand's outer contour.

It is the specific functioning and assumptions underlying each of the system components in Figure 2.1 that uniquely characterises each database driven system. A more detailed discussion of the aforementioned systems will now be presented, commenting specifically on the functioning of each of these specific subsystems.

3.4 Candidate parameter sets

The *potential* accuracy of database driven pose estimation systems is determined by the candidate parameters sets. Each possible set of parameters corresponds to one entry in the database. The more elements in the database, the higher the potential accuracy of the system. However, more entries imply the need for more memory, and for search procedures that can resolve queries with sufficient speed.

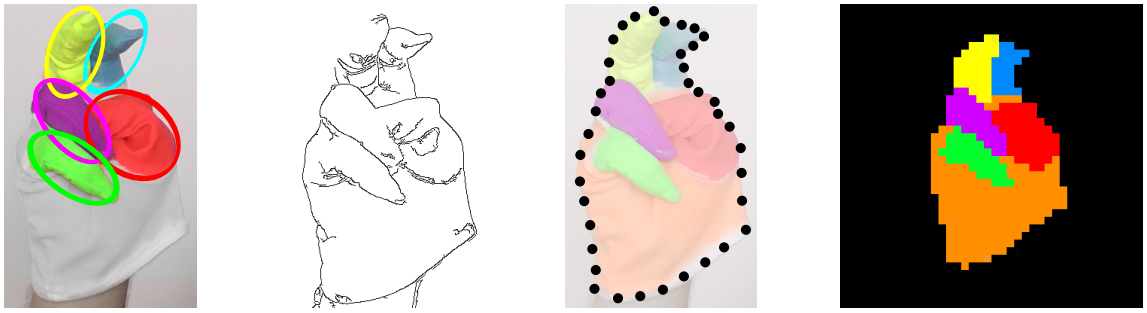
The systems in Athitsos et al. [6–10] vary in the number of database entries. The most recent case presented in [10] is intended for the *recognition* of American Sign Language. The system is, therefore, geared towards hand shapes that commonly occur during the signing of this specific language. A total of 20 such hand shapes were chosen. Each of these shapes is considered from 4032 different viewpoints in the final database, implying a total of 80640 database entries.

Zieren [91] presented a hand pose estimation system intended for use in a sign language tutor. As such, the system attempts to reconstruct arbitrary hand shapes. To do this, each finger was allowed to assume a discrete set of possible poses, with the thumb, index finger, middle finger, ring finger and pinky fingers having 7, 9, 8, 9 and 9 settings respectively. Therefore, a total of $7 \times 9 \times 8 \times 9 \times 9 = 40824$ hand shapes can be recognised by the system. This number was reduced to 23352 by eliminating physiologically improbable combinations of finger poses. Each hand shape was considered from 105 possible viewpoints. Thus, the database included $23352 \times 105 = 2451960$ postures.

The database elements need not necessarily be defined explicitly. The database of candidates in Grauman and Darrell [34] contained a set of 136500 randomly generated body pose parameter sets.

Wang and Popović [80] use a set of 18000 hand pose parameters captured using a data glove which include poses from fingerspelling[2], "common" hand gestures and random finger motions. They use the root mean square (RMS) distance between corresponding vertices of the synthetic model to sample a smaller subset of these pose parameters such that the elements of the subset are maximally separated. This subset of pose parameters is used to generate synthetic hand images, each rendered from multiple viewpoints, with a final database of 100000 elements.

[2]Sign language hand poses representing the alphabet of a given oral language.



(a) (b) (c) (d)

Figure 3.1: Features on which the existing systems operate: (a) the bounding ellipse features similar to those used in [91] with the image edited to resemble the glove used in that publication. (b) Canny edge-detector features used by Athitsos et al. [6–10]. This system used an *unadorned* and not a gloved hand. (c) Silhouette features of the type employed by Grauman and Darrell [34]. This system extracted silhouettes from whole body poses. (d) "Tiny image" features used by Wang and Popović [80]. The segmented input image is resized to a 40×40 raster, which forms the final features. This system employed a different glove design to that shown here.

3.5 Synthetic model

The 3D model used for rendering the pose parameters has a strong influence on the types of input images that may be processed.

Both Athitsos et al. [10] and Grauman and Darrell [34] made use of the commercially available Poser 5 [23] to render, respectively, an unadorned human hand and a full human body in various poses. As such, their systems are more generally applicable, but this has negative implications for the information available to the pose estimation system.

If a system makes special requirements of the user, such as requiring a coloured glove, then the synthetic model must reflect this. Zieren [91] made use of an OpenGL model of a gloved hand adorned in the same manner as required of the user. Each finger is coloured uniquely, and a marker is also placed on the back of the hand. While such systems are not generally applicable, they are useful when controlled environments can be reasonably expected. The additional information and better uniformity of the input data has positive implications for the accuracy of these pose estimation systems.

In the case of Wang and Popović [80], the glove design is produced algorithmically by choosing 20 seed triangles that are maximally separated in the synthetic glove model mesh. The seed triangles are randomly assigned one of a set of ten colours, and the rest of the model mesh triangles are coloured based on their nearest neighbouring seed triangle. The glove is then manufactured based on the results of this process.

3.6 Feature extraction

The feature extraction phase determines the type of information the system uses. All the systems discussed here make use of some form of contour features, though they are represented in different ways. The features used by each system are illustrated in Figure 3.1.

Zieren [91] approximated the different coloured markers on a user's glove using bounding ellipses. This served as a set of approximate contour features (each contour is associated with either a finger, or the back of the hand, based on the colour of the inside of the contour).

The system in Athitsos et al. [10] extracts edge images using the Canny edge detector [21]. The edge images themselves form the features on which the system operates.

Grauman and Darrell [34] extracted the silhouettes of the images of interest, and represented them as approximating point sets.

Wang and Popović [80] used the segmentation of the input image into marker regions directly by resizing the segmented image to a resolution of 40×40. The resulting images are referred to as "tiny image" features.

3.7 Database index generation and lookup

The key components of database driven pose estimation systems are the database index generation and lookup subsystems. Database index generation determines how the data in the database is structured, so that runtime queries can be processed with sufficient speed. Ultimately, the design of this subcomponent depends on the set of features extracted from input images, and the dissimilarity measure defined between them.

Because the rendering of a synthetic pose candidate is too processor intensive to occur at runtime, the features must be precalculated. The average size of the set of features associated with a synthetic image, along with the number of entries, determines the *size* of the database.

The *structure* of the database is dependent on the properties of the dissimilarity measure used to compare candidates. Large portions of the database can be eliminated quickly during lookup if a dissimilarity measure with convenient mathematical properties is chosen.

Since the bounding ellipses are especially simple features, Zieren [91] stores these features directly as was illustrated in Figure 3.1a. The Hausdorff distance [89] is then used as a distance measure to compare the input with database elements. Key concepts underlying this metric are illustrated in Figure 3.2. The Hausdorff distance is defined between two sets of points, in this case the sets of points defined by two ellipses. It can be interpreted as the maximum distance an adversary can force one to walk to get from either point set to the other. If A and B are the sets of points belonging to the two respective ellipses, then the Hausdorff distance between the sets A and B is

$$H(A,B) = \max(\max_{a_1 \in A} \min_{b_1 \in B} d(a_1,b_1), \max_{b_2 \in B} \min_{a_2 \in A} d(a_2,b_2)).$$

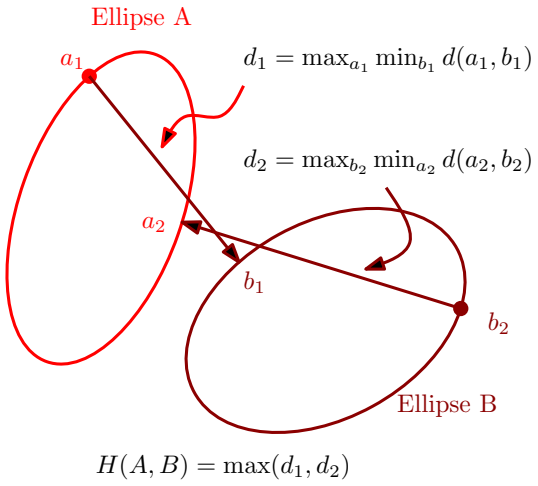


Figure 3.2: Illustration of the Hausdorff distance $H(A,B)$ between the ellipses A and B.

Note that the sets A and B can be chosen arbitrarily, provided that the distance measure d operates on all their elements. In this case, they are the points on an ellipse, but more general contours or point clouds are also a possibility.

In Zieren [91], d is chosen as the Euclidean distance between two points (making H a metric by virtue of d being a metric [89]). The database entries are stored in a tree structure, with similar elements stored near each other in the tree. The distance measure employed during the search procedure is the sum of the squares of the Hausdorff distance between the individual ellipses representing each finger's marker and the marker on the back of the hand.

When the dissimilarity measure is more computationally complex, using it directly is not feasible. Typically, some means of approximating the measure must be provided. This is often done by embedding the features into a lower dimensional space, where an approximate measure may be used to compare candidates.

The systems in Athitsos et al. [6–10] use (in various stages of development) the symmetric chamfer distance as the core dissimilarity measure. In its asymmetric form [8], the chamfer distance is given by

$$d_{AC}(X,Y) = \frac{1}{|X|} \sum_{x \in X} \min_{y \in Y} ||x - y||.$$

Here, X and Y are the two sets of edge contour points obtained from edge detection on the two hand images being compared. $|X|$ is the number of edge contour points in X, and $||x-y||$ is the Euclidean distance between a pair of points from X and Y. Essentially, the asymmetric chamfer distance is the normalised sum of the minimum distances from each point in X to some point in Y. Figure 3.3 illustrates the type of features expected by this metric, and the matches between the point sets X and Y involved in calculating d_{AC}. For the examples given in the figure, the asymmetric chamfer distances are given by

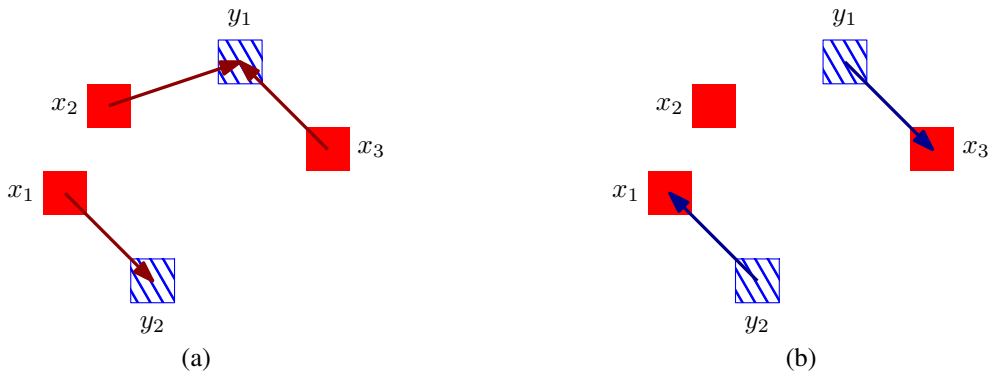


Figure 3.3: The matches involved in calculating the asymmetric chamfer distance are illustrated in (a) and (b) for $d_{AC}(X,Y)$ and $d_{AC}(Y,X)$ respectively, where X is the set of red filled pixels, and Y is the set of blue hatched pixels. These sets of pixels represent the edge features from different images, as illustrated, for example, in Figure 3.1b. After finding the closest matches in the target set for each point in the source set, all the distances represented by the arrows are added, and then divided by the number of points in the source set.

$$d_{AC}(X,Y) = \frac{1}{3}\left[d(x_1,y_2) + d(x_2,y_1) + d(x_3,y_1)\right]$$

and

$$d_{AC}(Y,X) = \frac{1}{2}\left[d(y_1,x_3) + d(y_2,x_1)\right].$$

For use in their work, Athitsos et al. define a symmetric form of the chamfer distance as

$$d_{SC}(X,Y) = d_{AC}(X,Y) + d_{AC}(Y,X).$$

In Section 4.2.3, the relationship between the chamfer distance and the EMD will be discussed. It will also be noted that the chamfer distance is susceptible to noise pixels, as a single noise pixel can match to any nearby pixels in the synthetic point set. The EMD resists this by keeping account of all matches, and ensures that small regions only have minor influence on the final result.

Because the chamfer distance is not a true metric, Athitsos et al. use Lipschitz embeddings to embed each database element into a normed space. This involves choosing N reference images I_n from the database. The Lipschitz embedding of the edge image X is then the vector

$$L(X) = \left[d_{SC}(X,I_1), d_{SC}(X,I_2), \ldots, d_{SC}(X,I_N)\right].$$

The underlying intuition is that two images X_1 and X_2 which are similar will have similar distances $d_{SC}(X,I_n)$ from the reference images, and so will have similar embeddings. This representation has the advantage that the embedding is a vector, and so all search methods applicable to vector spaces are relevant. In addition, the embeddings of all the synthetic images

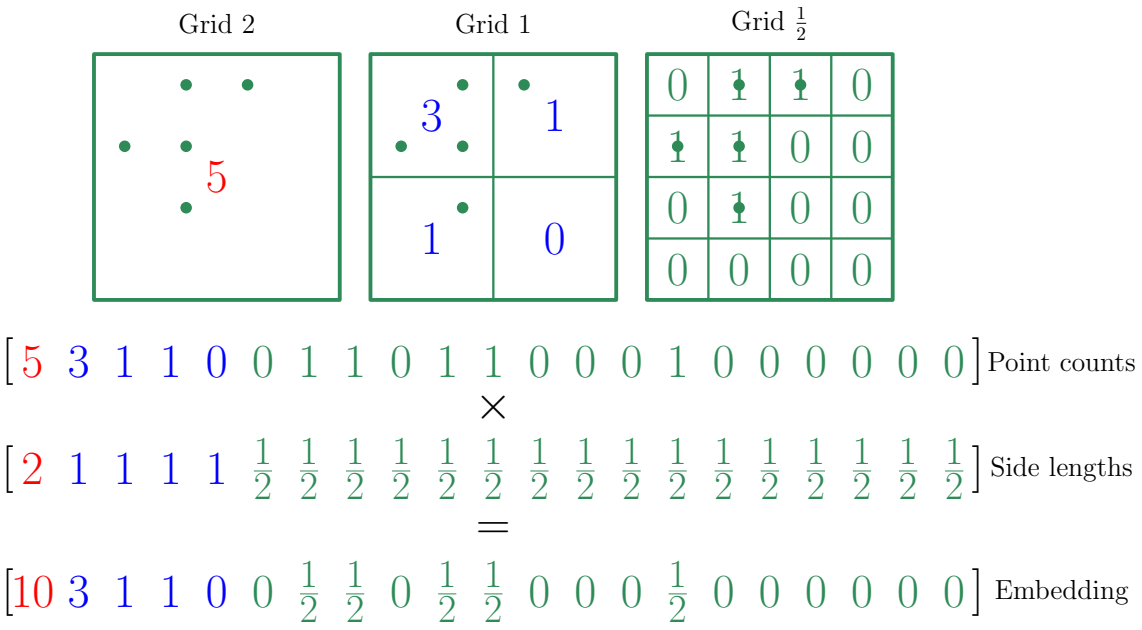


Figure 3.4: An illustration of the L_1 embedding of the EMD due to Indyk and Thaper [38]. Given a set of points with discretised locations, a vector is formed by concatenating the cells of progressively finer grids. Each cell contains the number of points within that cell, and each grid is finer than its predecessor by a factor of 2. Each cell is scaled by its side length, where the finest level has a side length of $\frac{1}{2}$. This embedding is employed by Grauman and Darrell [34] to embed silhoutte point features, similar to Figure 3.1c, into L_1.

can be precalculated, and the chamfer distance only needs to be calculated between the image and the N reference images to obtain the input image embedding.

Athitsos et al. select the reference images I_n by using a method of their own development called BoostMap which adapts ideas from AdaBoost [32] to the context of embeddings.

In Grauman and Darrell [34], the features are point sets extracted from the silhouettes of body poses (either real or synthetic). They apply an embedding of the EMD into L_1, due to Indyk and Thaper [38], who also provide theoretical guarantees on the degree of distortion induced by the embedding. The embedding is illustrated in Figure 3.4. Because the distance metric is now the L_1 (Manhattan) distance $d(\mathbf{v},\mathbf{w}) = \sum_n |v_n - w_n|$ locality-sensitive hashing [62] can be used to speed approximate query resolution.

The distance measure used by Wang and Popović [80] is similar to the symmetric chamfer distance, except the squares of the distances between points are added, provision is made for multiple region types, and the square root of the final result is taken. The measure penalises at each pixel of the image according to how distant the nearest pixel of the same type is in the other image. An asymmetric version of the "tiny image distance" (denoted by d_{AT} here) is defined first

$$d_{AT}(\mathbf{r}^i,\mathbf{r}^j) = \sqrt{\frac{1}{|\mathscr{C}_i|}\sum_{(x,y)\in\mathscr{C}_i}\min_{(u,v)\in\mathscr{S}_{xy}}[(u-x)^2+(v-y)^2]}$$

where $\mathbf{r}^i$ and $\mathbf{r}^j$ are tiny image features. $\mathscr{C}_i$ is the set of non-background pixels in the tiny image $\mathbf{r}^i$. $\mathscr{S}_{xy}$ is the set of tiny image pixels in $\mathbf{r}^j$ which have the same value as the pixel at (x,y) in the image $\mathbf{r}^i$. The tiny image distance is then defined as a symmetric version of d_{AT}, exactly as it is done in [8]

$$d_{ST}(\mathbf{r}^i,\mathbf{r}^j) = d_{AT}(\mathbf{r}^i,\mathbf{r}^j) + d_{AT}(\mathbf{r}^j,\mathbf{r}^i).$$

As this dissimilarity measure is non-trivial to calculate, binary codes are generated for each tiny image based on a method proposed by Torralba et al. [68] which maps similar tiny images to binary codes which are close in terms of their Hamming distance. The hand pose estimation system employs a search using the Hamming distance, followed by a refinement step using the tiny image distance d_{ST}.

3.8 Conclusion

This chapter has presented an overview hand pose estimation, and in particular a set of closely related database driven pose estimation systems. Types of systems discussed include model-based trackers and a variety of single frame pose estimation systems.

Model-based hand trackers exploit time coherence between successive frames of video to track changes in hand pose. However, the speed and unpredictability of hand motions during signing undermines these time coherence assumptions, rendering model-based trackers less suitable for application in sign language tutors. Single frame pose estimation systems do not rely on time coherence assumptions, and so can more readily deal with such rapid changes. Techniques for performing single frame pose estimation include inverse kinematics, learning mappings of features to pose parameters, forming classification cascades and database-driven search for pose candidates.

Inverse kinematic techniques rely on being able to identify relatively high-level information about the hand, such as the location of the fingertips, which are challenging to identify when compared with low-level features such as edges or coloured regions. In addition, a good estimate of the hand pose must be available, such as in Wang and Popović [80], or it must be fixed as in [50]. The reliance on higher-level features and pose information are obstacles to employing inverse kinematic techniques *in isolation* for a tutor system.

While promising, learning mappings of input features to parameters and classifier cascades created using boosting techniques may suffer from scarcity of real training data. If the hand pose is allowed to vary freely, providing sufficient amounts of real images with ground truth data to adequately cover the state space is challenging. However, learning synthetic data introduces the risk of overfitting on spurious regularities in the synthetic data. At the same time, gathering of sufficient test data is hampered by the same state space size considerations as in the collection of training data.

Instead, using a fixed but perceptually meaningful metric has no risk of overfitting.[3] This is the approach employed by database-driven methods. While concerns about the accuracy of the model used to generate the database remain, there is no opportunity for spurious regularities in the synthetic model to become deciding factors in the estimation process.

Several existing database-driven systems were discussed. Athitsos et al. [8] and Grauman and Darrell [34] employ markerless tracking, but this unnecessarily lowers the amount of information available to the tracker when, for sign language tutors, it is reasonable to expect a user to wear coloured gloves.

Coloured gloves enable simple, yet powerful, similarity searches. In particular, being able to separate the finger markers immediately leads to perceptually meaningful and rapidly evaluated, metrics which compare the approximate appearance of corresponding fingers. Zieren [91] provides an example of such a simple marker-based metric. However, used in isolation, such metrics are not able to consider detailed hand shape.

The glove-based approach employed by Wang and Popović [80] is promising, though questions for further investigation arise. Spatially separated regions of the glove have similar colouring. While it is possible that the exact configuration of all coloured regions may be a sufficiently strong indicator of the true hand pose, the possibility that identically coloured regions act as mutual confusers needs to be addressed. In addition, the distance measure uses searches for nearest pixels which may be sensitive to segmentation noise.

The hand pose estimation system presented in this dissertation makes use of coloured gloves to take advantage of the greater environmental control afforded to tutor systems. A glove similar to that of Zieren [91] is adopted. However, a more detailed form of similarity metric, the EMD, will be employed. While being more informative than the Hausdorff-distance metric employed by Zieren [91], the EMD is also less sensitive to noise pixels, as opposed to chamfer distance type dissimilarity measures such as employed by Athitsos et al. [8] and Wang and Popović [80].

In the next chapter, the theoretical underpinnings of similarity search in metric spaces will be presented in more detail and the EMD will be defined formally. These techniques will subsequently be employed to develop the tutor's hand pose estimation system.

[3]The choice of a "perceptually meaningful" metric could be construed to be the avoidance of overfitting in the sense of consciously avoiding brittle metrics. For example, the use of the Euclidean distance on the vector of image pixels is usually a poor choice in that it does not take into account similarity considerations appropriate to images.

Chapter 4

Similarity Search in Metric Spaces

Metric spaces [89] are a powerful tool for performing similarity search within databases that contain vast numbers of candidates. These spaces preserve the familiar concept of distance from geometry while generalising it to include a much greater variety of objects as possible points in space.

In this chapter, the properties of metric spaces and how these relate to similarity search are discussed. Theoretical matters which are common to all application domains are considered first, with the focus gradually shifting to human body/hand pose estimation, in particular using the Earth Mover's Distance (EMD). The functioning of exact search methods will be outlined, after which methods for accelerating and approximating these exact techniques will be considered.

Formally, a metric space is an ordered pair (S,d), where S is a set of objects of interest (analogous to points in space), and d is a distance measure defined between objects in S. The distance measure d must exhibit the following properties (where x, y and z are elements of S) [89]:

$$\begin{aligned} d(x,y) &\geq 0 && \ldots \text{non-negativity} \\ d(x,y) &= d(y,x) && \ldots \text{symmetry} \\ d(x,y) &= 0 \iff x = y && \ldots \text{identity} \\ d(x,z) &\leq d(x,y) + d(y,z) && \ldots \text{triangle inequality.} \end{aligned}$$

Together, these properties preserve key notions central to intuitions regarding distance. A distance measure that satisfies all of these requirements is called a metric.

The first property ensures that distances are always positive, while the second property requires that the distance from one point to another is the same as that of the return journey. The third property states that the distance between two points is zero if and only if they are the same point. The fourth property guarantees that the distance of a route between two points that goes via a third point is always at least as large as the direct route between the two endpoints.

The Euclidean distance in $\mathbb{R}^n$ is an obvious example of a distance measure that exhibits these properties. It is important to note, however, that there is nothing in these properties that constrains the nature of the objects in S to be vectors. They may be documents or sets of points or images, or any other domain of interest. This is one source of the flexibility of metric spaces as a tool for similarity search.

4.1 Similarity search in databases

Nearest neighbour search is a simple yet powerful technique for finding candidates most similar to a given item amongst a large number of potential matches stored in a database $D \subset S$.

Given some measure of dissimilarity d, one can, in principle, determine the distance between the query object x and all possible candidates $x' \in D$, retaining those candidates for which $d(x,x')$ is smallest.

However, if there are no restrictions placed on d, this strategy quickly becomes impractical as the number of potential candidates grows. This is particularly evident when the distance measure d is computationally expensive. As there is no way of knowing even approximately what d will be until it is calculated in each case, all items need to be considered.

Requiring that d be a metric enables search strategies which are able to discard large portions of a database based on distances calculated between the query object and selected items in the database called pivot objects. The selection of pivots and the manner in which these distance calculations are used to eliminate candidates is specific to the algorithm employed.

4.1.1 Vantage point trees

As an example of such a search strategy, the vantage point tree (or VP-tree) [85] will be considered. Essentially, VP-trees are a kind of binary space partitioning scheme, but generalised to metric spaces.

A key concept employed in this scheme is that of a ball partition. Given some pivot object p, the set of database elements D can be divided into sets S_i and S_o by specifying a radius d_m such that S_i are all the elements closer than d_m to the pivot and S_o all those elements further than d_m from the pivot.

$$S_i = \{o \in D \cdot d(o,p) \leq d_m\}$$

$$S_o = \{o \in D \cdot d(o,p) \geq d_m\}.$$

Choosing the radius d_m to be the median of the distances $d(o,p)$ ensures that the two sets are equal in size (objects on the dividing radius are shared between the partitions for this reason).

This type of partitioning is referred to as a ball partition, because it is analogous to dividing points in $\mathbb{R}^3$ into two regions using a spherical surface centred at p with radius d_m. This idea is illustrated in Figure 4.1.

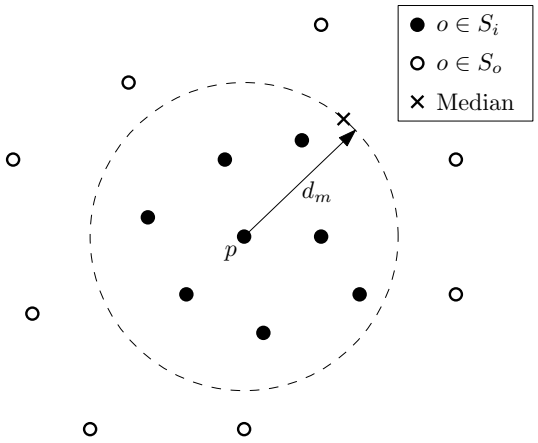


Figure 4.1: Conceptual view of a ball partitioning. Note that the median element is shared between the two partitions.

A vantage point tree consists of a binary tree where each node contains a pivot (also known as vantage points), usually selected by means of some heuristic. At the root, a ball partition is made based on the root pivot, and the two partitions are each assigned as a subtree. This process continues recursively until each leaf has less than a predetermined number of elements.

Given a query object q, one can find nearest-neighbours for q by recursively descending through the binary tree. At each node, the distance from the query object to the pivot is calculated. Using the distance from the query object to the pivot $d(q,p)$ and the distance from the pivot to the median object d_m, various constraints can be imposed on the distance between the query object and any object o in either S_i or S_o. For example, from the triangle equality it can be shown that [89]

$$|d(q,p) - d(p,o)| \leq d(q,o) \leq d(q,p) + d(p,o).$$

This is known as the object-pivot distance constraint. The constraint is illustrated in Figure 4.2. The right-hand restriction is derived from the case where $d(q,p)$ and $d(p,o)$ reinforce each other such that the path from q to o via p is the same length as the path directly from q to o. The left-hand constraint emerges in the case where $d(q,p)$ and $d(p,o)$ work against each other.

Other constraints also naturally emerge from the properties of the metric d, some involving both parent and child pivots. These constraints can be used to eliminate large portions of the database using a minimal number of distance calculations.

Having discussed the general principles underlying similarity search in metric spaces, the key metric of interest to this dissertation, the EMD, will now be presented.

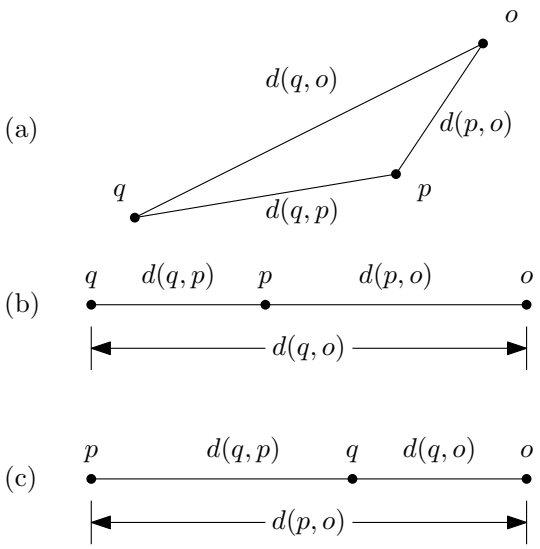


Figure 4.2: Visualisation of the object-pivot distance constraint: (a) Non-extreme case: $d(q,o) \leq d(q,p)+d(p,o)$ (b) Extreme case: $d(q,o) = d(q,p)+d(p,o)$ where $d(q,p)$ and $d(p,o)$ reinforce each other (c) Extreme case: $|d(q,p)-d(p,o)| = d(q,o)$ where $d(q,p)$ and $d(p,o)$ partially cancel each other.

4.2 The Earth Mover's Distance

4.2.1 Introduction

The EMD is, intuitively, the minimum amount of work required to redistribute M piles of earth p_i with volumes w_{p_i} to fill N holes q_j with volumes w_{q_j}. The distances between the landmarks p_i and q_j are given by $\lambda(p_i,q_j) = \lambda_{ij}$, referred to as the ground distance. The $M \times N$ flow matrix F has elements f_{ij} which represent the amount of earth moved from pile p_i to hole q_j.

A set P of weight/landmark pairs is referred to as a signature. More formally, a signature is a set P of M ordered pairs (w_{p_i}, p_i), where $i \in 1 \ldots M$.

The EMD between a signature P and signature Q is defined as

$$d_{\text{EMD}}(P,Q) = \min_F \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} \lambda_{ij} f_{ij}}{\sum_{i=1}^{M}\sum_{j=1}^{N} f_{ij}} \tag{4.2.1}$$

where f_{ij} are elements of the flow matrix F, subject to the conditions [57]

$$\begin{aligned} f_{ij} &\geq 0 \\ \textstyle\sum_{j=1}^{N} f_{ij} &\leq w_{p_i} \\ \textstyle\sum_{i=1}^{M} f_{ij} &\leq w_{q_j} \\ \textstyle\sum_{i=1}^{M}\sum_{j=1}^{N} f_{ij} &= \min(\textstyle\sum_{i=1}^{M} w_{p_i}, \sum_{j=1}^{N} w_{q_i}). \end{aligned}$$

These conditions respectively ensure that flow is positive, that the outward and inward flows are limited by the weight attached to a given landmark, and that maximum total allowable flow between the signatures occurs.

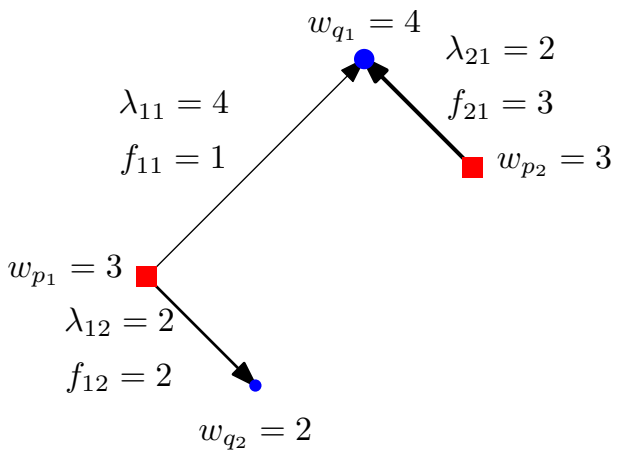


Figure 4.3: Example of an EMD match between two signatures. The source signature is represented by red squares, and have equal weights of 3 each. The sink signature is represented by blue circles, and have weights of 4 and 2 respectively.

The case where the signatures P and Q have equal total weight ($\sum_i w_{p_i} = \sum_j w_{q_j}$) is of particular interest. This is due to the fact that, if these are equal, and the ground distance λ is a metric, then the EMD is also a metric [57], and so can be used by metric space similarity search techniques. In the discussion that follows, $\lambda(p_i, q_j)$ is chosen to be the Euclidean distance between landmarks p_i and q_j.

An example of a possible EMD match for two signatures is shown in Figure 4.3. Here, the source signature is represented by two red squares with an equal weight of three. The sink signature is represented by blue circles, with the top circle having a weight of four, and the bottom having a weight of two.

In the illustration, the top circle in the sink signature absorbs all the earth from the top square, and one unit of earth from the bottom square. The other two units of the bottom square are assigned to the bottom circle.

The EMD, given this flow, is then the sum of the products of the flow and distance associated with each of the arrows, normalised by the sum of all the flows

$$\begin{aligned} d_{EMD}(P,Q) &= \frac{f_{11}\lambda_{11} + f_{12}\lambda_{12} + f_{21}\lambda_{21}}{f_{11} + f_{12} + f_{21}} \\ &= \frac{1\times 4 + 2\times 2 + 3\times 2}{1+2+3} = \frac{14}{6}. \end{aligned}$$

What is immediately apparent from this example, is that the EMD favours configurations of points that are geometrically similar. Minor displacements in the configuration of points have little effect, as the flow required to move these misaligned points is scaled by the distance travelled which, if the displacement was indeed small, will have a minimal overall effect.

It is this property that makes the EMD useful as a measure of dissimilarity between configurations of points.

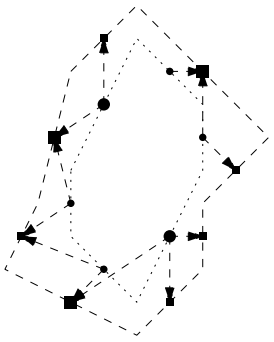


Figure 4.4: Two hypothetical contours, shown by dotted and dashed lines, and their possible EMD matching indicated by the arrows. The weighted points approximating each contour's segments are indicated by circles in the "source" contour and squares in the "sink" contour, with larger sizes indicating larger point weights for longer segments.

4.2.2 Contour-based Similarity Search

Because the EMD is useful for establishing whether configurations of points are geometrically similar, it can also be used to measure the similarity between pairs of *contours*.

As discussed in Section 3.6, Grauman and Darrell [34] used the EMD to compare the silhouette contour points of object pairs. Sampled points along the contour were collected into a signature of equally weighted masses, suitable for use with the EMD.

In this dissertation, a similar approach is taken in generating EMD signatures from marker contours. However, instead of using equally weighted points, the segments of the piecewise continuous contours are replaced by point masses at the segment centroids, with weights directly proportional to the segment length. The segment weights are scaled so that the total weight of a given signature equals one.

The use of weights proportional to the segment length is important when considering how the contour extraction algorithm might produce slightly different discretisations of the same contour portion. In particular, consider the case of a single segment in one signature being divided in half within the corresponding contour portion of the other signature. Clearly the two half segments should both match to the original segment. Therefore, the weight of the half segments is set to half that of the original segment, so that the "earth" representing the original segment may be divided perfectly to fill the "holes" representing the half segments. Without this weighting, the half segments would need to source "earth" from another source in order to be filled, which would represent poor matching behaviour.

The EMD matching of two such contours is illustrated in Figure 4.4. The segment centroids are shown as circles and squares respectively, and a possible solution for the flow of mass between them is indicated by the arrows.

Because several sets of contours are present (one for each of the six markers), this process results in six EMD signatures per hand image. The distance d between any two hand configurations h and h' is taken to be the sum of the EMD between the corresponding marker signatures

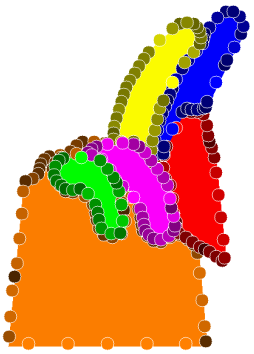


Figure 4.5: Example of an EMD signature extracted from a synthetic image.

s_k and s'_k, with $k \in \{1 \ldots 6\}$:

$$d(h,h') = \sum_{k=1}^{6} d_{\text{EMD}}(s_k, s'_k). \tag{4.2.2}$$

This will be referred to as the compound EMD. Note that this is not the form used by existing systems such as [34], which use d_{EMD} directly. Appendix A contains a proof that this is a metric by virtue of d_{EMD} being a metric.

When a marker is completely occluded, it is replaced by a small dummy segment at the origin.

Using the compound EMD, a K-nearest neighbour search may be used to find pose candidates for an input image within a set of synthetic pose candidates.

An example of a compound set of EMD signatures obtained from a synthetic hand image is given in Figure 4.5 to give an indication of the complexity of the task.

4.2.3 Relation to the chamfer distance

Recall that, in its asymmetric form, the chamfer distance is given by

$$d_{AC}(X,Y) = \frac{1}{|X|} \sum_{x \in X} \min_{y \in Y} ||x - y||$$

Here, X and Y are the two sets of edge contour points of the two hand images being compared. $|X|$ is the number of edge contour points in X, and $||x-y||$ is the Euclidean distance between a pair of points from X and Y.

It should be noted that the asymmetric chamfer distance is a non-metric special case of the EMD, where the source signature has landmarks with equal weights, and the sink signature has weights approaching infinity. This allows each landmark in the source signature to match to the closest landmark in the sink signature without any interaction with other source landmarks. Thus, the use of the EMD in this work can be seen as a natural extension of Athitsos et al. [8].

Figure 4.6 contrasts the matching behaviour of the chamfer distance (asymmetric case) with that of the EMD. From Figures 4.6a and 4.6b, it can be seen that even in the absence of noise, the chamfer distance is more prone to losing shape information due to mismatched points.

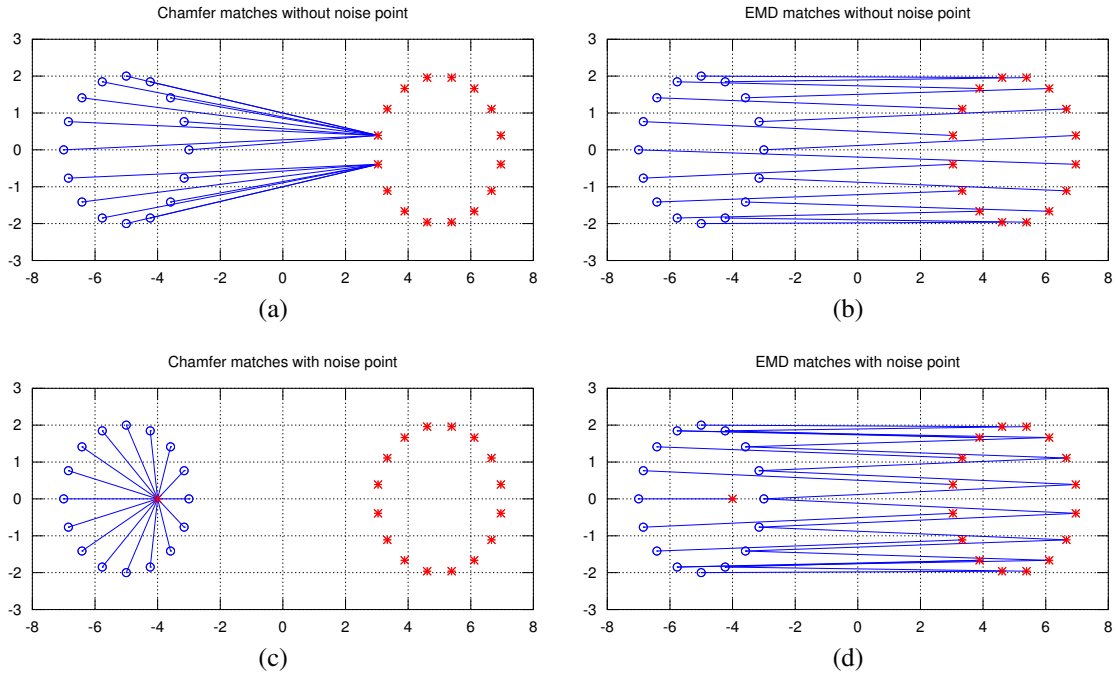


Figure 4.6: Contrastive examples of the point matches underlying the asymmetric chamfer distance and the EMD. The shape preserving tendencies of the EMD are demonstrated by (a) and (b). The EMD tolerance to noise points is demonstrated by (c) and (d). EMD match figures show matches accounting for at least 90% of the total flow of mass per point.

The chamfer distance is also sensitive to noise, because a single noise point acts as an infinite sink, causing all nearby source pixels to mismatch. The EMD remedies this by allowing a single pixel to have only finite influence on the signature as a whole. Figures 4.6c and 4.6d contrast the behaviour of the chamfer distance and the EMD in this regard.

4.3 Summary

In this chapter, similarity search using the properties of metric spaces were discussed. In particular, the principles underlying VP-trees were discussed. This data structure will be used as a baseline with which exact queries can be executed, so that the accuracy of the approximate method presented in this part of the dissertation may be evaluated.

The EMD was introduced as a metric which is particularly applicable to determining the geometric similarity of point sets and, with minor modifications, sets of contours.

In addition, its relationship with the asymmetric chamfer distance as used by Athitsos et al. 4.2.3. It was demonstrated that the EMD both preserves shape information and resists noise to a greater extent than the asymmetric chamfer distance, making it a promising substitute.

In the next chapter, the specifics of the pose estimation system are presented, which uses the compound EMD introduced in Equation 4.2.2 as a dissimilarity metric.

Chapter 5

Experimental Setup

In this chapter, the system used to test the two-tier approximation scheme proposed in this dissertation is presented. The aim is to test the accuracy of the compound EMD metric proposed in Equation 4.2.2 as a means of estimating hand poses from still images. An exact query database using a VP-tree will be used to generate these results.

The results obtained from exact query resolution are then compared with the two-tier approximation scheme, which will be described in this chapter. A comparison with the performance using the symmetric chamfer distance is also provided to place the results within the context of existing literature.

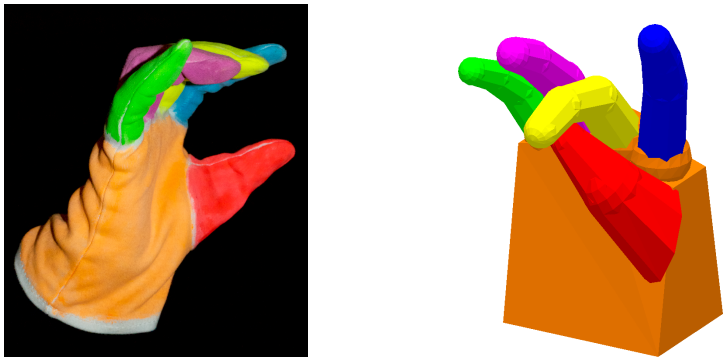
Note that, as it is the metric under investigation, and since the discussion of metrics in general has drawn to a close, the compound EMD is simply referred to as d from this point onward. The components of the system will now be presented.

5.1 Hand model

The glove employed for this experimental evaluation is pictured in Figure 5.1, along with a rendering of the 3D model thereof (the gloves used in the final system are based on this design). The glove is similar to that used by Zhang et al. [90] and Zieren [91], but has a coloured palm, and forgoes the marker on the back of the palm. Candidate poses are rendered synthetically using OpenGL. Images were rendered with a resolution of 256×256. Feature extraction is then performed on the synthetic image.

Each finger is allowed to assume a discrete set of poses. These were chosen to be similar to those proposed by Zieren [91]. Separately, the thumb, index-, middle-, ring- and pinky fingers could assume 7, 9, 8, 9 and 9 poses respectively. Therefore, for a given viewpoint, a total of $7 \cdot 9 \cdot 8 \cdot 9 \cdot 9 = 40824$ poses are possible.

As expected, exact search operations using the compound EMD were extremely slow. For this reason, the tests on the pose estimation system used a smaller set of possible viewpoints than the final system (about 10% of the final number). The number of possible hand shapes



(a) (b)

Figure 5.1: (a) The glove worn by the user and (b) the 3D model used to approximate it in generating the synthetic database.

is identical. After reduction of the number of viewpoints, exact search operations took several hours for each of the 500 test cases.

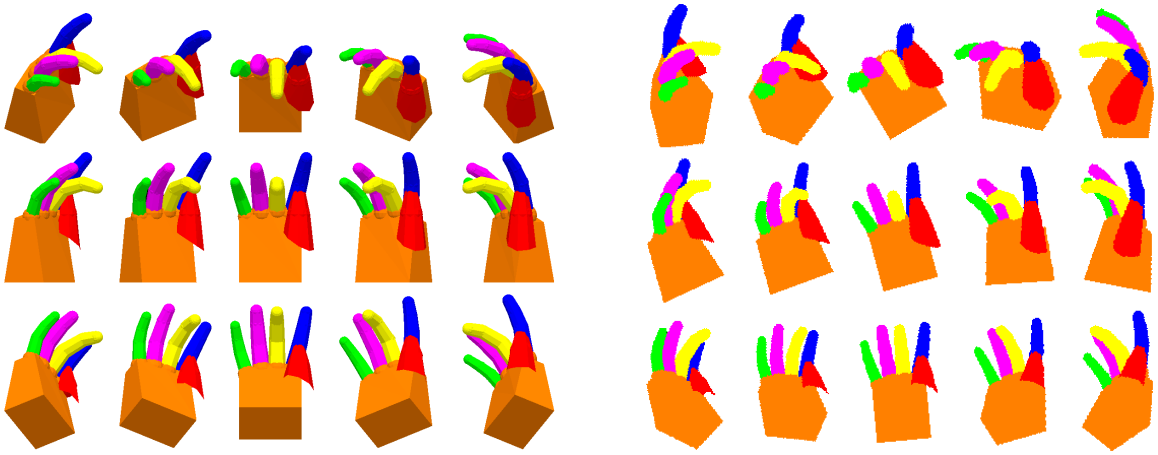
The set of 15 viewpoints represented in the database were centred around a frontal view of the inside of the palm. With the open palm aligned vertically with the y-axis, all azimuthal rotations $-60°$, $-30°$, $0°$, $30°$ and $60°$ were considered, with elevations $-45°$, $0°$ and $45°$. This is shown in Figure 5.2a.

Taking into account each viewpoint, the database contains $40824 \times 15 = 612360$ elements.

5.2 Image registration

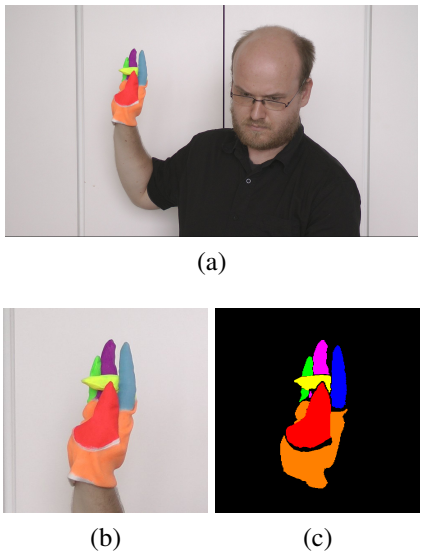
5.2.1 Processing of real test images

Test images, unlike synthetic images, need an initial processing stage to locate the hand and extract binary masks for each marker. A raw image from the test set is shown in Figure 5.3a.



(a) (b)

Figure 5.2: (a) Example pose rendered from all viewpoints represented in the database. (b) Masks for each case normalised for scale and rotation using the procedure detailed in Section 5.2.



(a)

(b) (c)

Figure 5.3: An example image from the test set is shown in (a). Colour likelihood models are used to detect marker pixels, after which the CAMSHIFT algorithm is used to locate the hand within the image. The bounding box obtained from CAMSHIFT is expanded and centred on the centre of mass of the marker pixels, resulting in (b). Maximum likelihood classification and thresholding are used to produce binary masks for each marker. The resulting classified regions are shown together in (c).

Colour likelihood models (one for each marker) are used to locate probable marker pixels within the image. The CAMSHIFT algorithm [16] is used to locate the hand based on the marker likelihoods. The bounding box obtained from the CAMSHIFT algorithm is then refined by enlarging it (as the initial bounding box tends to be too small) and centring it on the centre of mass of all the marker pixels combined. An example of an extracted bounding box region is shown in Figure 5.3b. Maximum likelihood classification and thresholding based on the likelihoods is used to obtain binary masks for each marker. Marker segmentation results are illustrated in Figure 5.3c.

5.2.2 Image normalisation

For both synthetic and real images, the origin is chosen to be the centre of mass of all the markers (fingers and palm) combined.

In order to standardise the marker masks to some scale and rotation, the system considers the two regions within the image occupied by the palm, and by the fingers respectively. It can be observed that, even when fingers occlude each other, the region of the image occupied by the fingers taken together remains fairly unchanged. The centre of mass of this region, along with that of the palm, is used to obtain a rough estimate of the hand's image-plane rotation θ_i. This is illustrated in Figure 5.4. As a further demonstration, the different viewpoints shown in Figure 5.2a are given along with their normalised marker masks in Figure 5.2b.

An estimate of the hand scale is obtained from the zero-order moments of the finger- and palm masks, represented by f_{00} and p_{00} respectively. The zero-order moment of a mask $M(i,j)$ is, by definition, $\sum_{(i,j)} M(i,j)$. As the masks are binary, the zero-order moments are the number

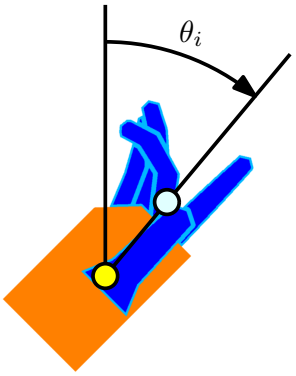


Figure 5.4: The procedure used to obtain the image-plane rotation θ_i of the hand as described in Section 5.2. The two circles represent the centroids of the palm and fingers respectively.

of non-zero pixels in each mask, and therefore proportional to the area of the non-zero regions. The scale of the hand is subsequently estimated as $l_i = \sqrt{f_{00} + p_{00}}$.

Rotating the masks by $-\theta_i$ and scaling by $1/l_i$ serves to standardise both to a single coordinate system which is fairly stable with respect to small changes in the input pose.

Note that, in the final system, the system explicitly tests 32 possible rotations for each pose candidate, and so this rotation step is not used. Testing for 32 additional rotations explicitly during exact searches would have been prohibitively expensive in terms of computer time, necessitating this approximation. The scaling procedure is still used in the final system.

The contours around the markers are extracted, and approximated using a form of the Teh-Chin contour approximation algorithm [67] available in the OpenCV library [17].

5.3 Search structures

In order to perform exact K-nearest neighbour search efficiently, VP-trees were employed. Query resolution takes the form of a series of distance comparisons using the compound EMD d.

However, to compare the input with synthetic pose candidates, the EMD signatures of the pose candidates must be available at runtime. Because the EMD signature of a pose candidate tends to be fairly large (the contours associated with one synthetic pose candidate contain, on average, 360 vertices in total), storing the signature of every candidate in advance may be impractical. This necessitates rerendering synthetic pose candidates at runtime (a further reason why the database size is limited in this part of the investigation).

Furthermore, the EMD is computationally complex, and so its runtime evaluation should be limited to as few instances as possible.

In response to these challenges, the following two-tier search approximation scheme is proposed. A rough search is performed using a more easily evaluated metric d^* to generate N hypotheses, which can then be sorted using direct evaluation of the EMD. Note that the rough search is an *exact* search, but the scheme is approximate because this search is using a different

metric in the place of d during this phase.

A compact set of features is chosen which may be precalculated for each candidate and stored within the index structure, similar to the approach of Zieren [91]. The selected features for a segmented hand image h are the areas a_k (in pixels) and centroids $\mathbf{c}_k = (x_k, y_k)$ of each of the $k \in \{1 \dots 6\}$ markers. The metric used for the rough search is defined between two segmented hand images h and h' as

$$d^*(h,h') = \sum_{k=1}^{6} ||\mathbf{c}_k - \mathbf{c}'_k|| + |\sqrt{a_k} - \sqrt{a'_k}| \tag{5.3.1}$$

where $||\mathbf{v}||$ is the Euclidean norm. The sum over the first term represents a rough approximation of Equation (4.2.2) calculated between the corresponding marker centroids (now masses with unity weight). The second term represents a basic measure of dissimilarity between the surfaces enclosed by the pairs of contours.

Equation (5.3.1) can be evaluated rapidly, especially if the centroids and areas have been precomputed. By performing the initial search with d^*, the two-tier search procedure has the virtue of limiting the EMD evaluations to a known number N. The experimental investigation of the tradeoff between speed and accuracy implicit in the choice of N is the subject of the next chapter.

Because the symmetric chamfer distance (d_{SC}) is not a metric, VP-trees could not be employed during nearest neighbour search. Therefore, a linear search through the set of possible poses was performed, so as to ensure the distance measure is used with full accuracy.

5.4 Summary

In this chapter, the characteristics of the system used to evaluate the two-tier approximation of the compound EMD were presented. These included the glove design, the synthetic model, the image processing methods and the database indexing methods employed.

In particular, a two-tier search method was proposed that uses a rough metric d^* during an initial nearest neighbours query for N objects. These N objects are then sorted using the exact compound EMD to produce the final ranking of the results. The metric d^*, during its calculation, performs a small set of operations on a compact set of features, the areas and means of each marker region of the glove. Thus it can be quickly evaluated, and the features of all the synthetic images can be precalculated and stored with relative ease.

The effectiveness of the different search methods are tested and discussed during the course of the chapter that follows.

Chapter 6

Experimental Evaluation

In this chapter, the two-tier search approximation to the compound Earth Mover's Distance (EMD) is evaluated as a means of hand pose estimation in still images. Comparisons will be made to the exact searches using the compound EMD, as well as the symmetric chamfer distance used by Athitsos et al. [8][1]. It is demonstrated that the two-tier search approximation outperforms exact search using the symmetric chamfer distance, and approaches that of the exact search using the compound EMD. Furthermore, it will be shown that a performance increase of two orders of magnitude is obtained using this approximation.

6.1 Test set

Images were recorded of a user wearing the glove in various poses, in the process generating 500 test images with ground truth data. Figure 6.1 provides an example of one such image. Ground truth data was generated and the pose displayed for the user to mimic. The recorded images and ground truth data are stored for system performance evaluation.

Figure 6.1: A sample image from the test set.

Ground truth data included finger joint angles, model azimuth and model elevation. As a compromise with respect to data collection feasibility, the user was not required to adopt a certain image-plane hand location, scale or rotation. These variables are not included within

[1]Note that the systems presented in [6–10] discuss systems that *approximate* the symmetric chamfer distance defined in [8], and so using the symmetric chamfer distance directly is most appropriate as a means of comparison.

ground truth data, but are implicitly obtained using the alignment procedure described in Section 5.2.

The model viewpoint and finger joint angles were chosen at random from continuous uniform distributions over the range of possible variation of each variable. The viewpoint azimuth varied between -72°and 72°, and the viewpoint elevation varied between -60°and 60°. The user was allowed to generate a replacement pose only if the current pose was too uncomfortable or physically impossible.

It should be emphasised that this ground truth data is inherently approximate, as the user may have made small errors in mimicking the pose. This may affect the maximum possible accuracy that a recognition algorithm can achieve relative to this information.

6.2 Nature of tests performed

Four approaches to similarity search were tested. A selection of test images are shown in Figure 6.2, along with the nearest neighbour obtained with each method superimposed on the original image.

To serve as a baseline, the first approach tested nearest neighbour search using an implementation similar to Athitsos et al. [8], using the symmetric chamfer distance d_{SC} as the dissimilarity metric operating on the entire set of hand contours. Image registration was, however, performed using the method described in Section 5.2. Example nearest neighbour results are displayed under “chamfer distance” in Figure 6.2.

The second approach tested the use of the rough metric defined in Equation (5.3.1) as an underlying metric for nearest neighbour search. Example nearest neighbour results are displayed under “Areas and Means” in Figure 6.2. Note that no use of the EMD was made in this case.

The third approach tested the result of direct nearest neighbour search using the compound EMD. Example nearest neighbour results are displayed under “Direct EMD” in Figure 6.2.

The fourth approach combined a nearest neighbour search using the area- and centroid-based distance metric with the compound EMD as a final refinement step. In the first step, a set of N nearest neighbours are returned using the metric in Equation (5.3.1). To determine the effect of N, results were obtained for $N = 32$, $N = 64$ and $N = 128$. The nearest neighbours are then sorted using direct evaluation of the EMD with respect to the input image. For $N = 128$, example nearest neighbour results are displayed under “Two-Tier” in Figure 6.2.

In order to evaluate the quality of the results, two error measures were used. Primarily, the difference between the position of the fingertips in the candidate pose and the ground truth pose were used as indicators of the quality of the match. For this measure, the 3D model of each hand pose is left unrotated regardless of the viewpoint. The locations of the fingertips in the unrotated 3D model are solely a function of the finger joint angles obtained from the ground truth pose and the estimated pose parameters. Therefore, this error measure is representative of hand shape error, decoupled from viewpoint error.

Figure 6.2: The leftmost column shows the hand as detected in the raw input image. The other columns show the nearest neighbour result obtained using different search strategies. The second column shows the results obtained using the baseline system which employs the symmetric chamfer distance d_{SC} (based on [8]). The third column considers only the marker areas and means (metric d^*). The fourth column shows the result of exact queries using the compound EMD. The fifth column shows results generated by the two-tier search which finds an initial set of 128 nearest neighbours using d^* and refines these using the EMD. (Continues on next page)

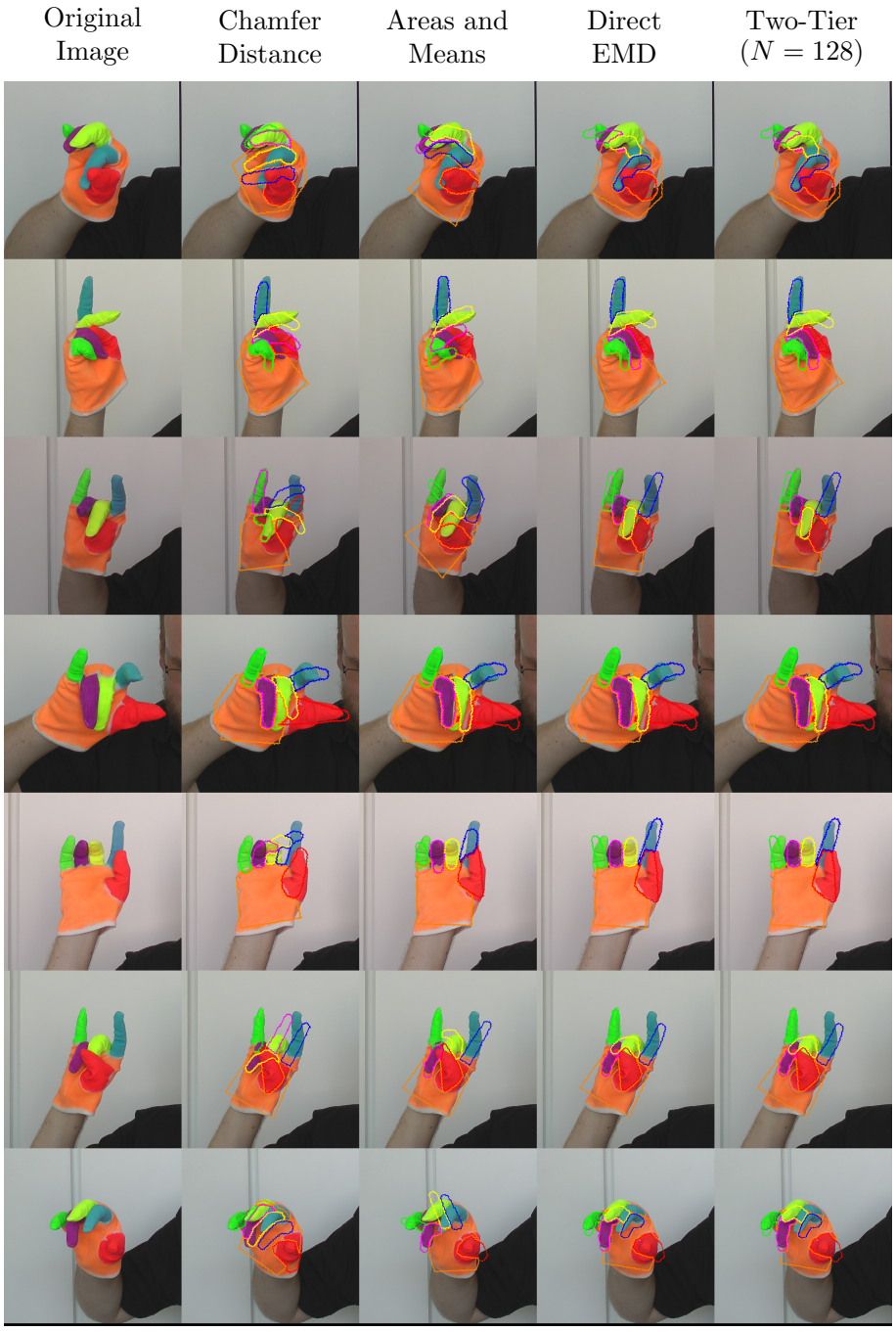


Figure 6.2: The leftmost column shows the hand as detected in the raw input image. The other columns show the nearest neighbour result obtained using different search strategies. The second column shows the results obtained using the baseline system which employs the symmetric chamfer distance d_{SC} (based on [8]). The third column considers only the marker areas and means (metric d^*). The fourth column shows the result of exact queries using the compound EMD. The fifth column shows results generated by the two-tier system which finds an initial set of 128 nearest neighbours using d^* and refines these using the EMD.

For single hypothesis tracking, only the closest neighbour found was considered. The average error over all fingertips in all test cases, Δ_p, is tabulated in Table 6.1. Average errors for the individual fingers are shown in Table 6.2. Lower bounds are obtained from the case where an oracle returns the database candidate with the least error.

When multiple hypotheses are allowed, an idealised value for obtainable accuracy may be found by selecting the candidate with the smallest error amongst the k nearest neighbours. Figures 6.3 and 6.4 show the behaviour of the average fingertip position error over all test cases and all fingertips for values of k ranging from 1 to 20. Note that, for $k = 1$, the values are identical to those in Table 6.1.

To measure the error in the global orientation of the hand, the angle through which the candidate palm must be rotated to align with the ground truth palm was used. The average Δ_ϕ of this angle over the nearest neighbour of each test cases is tabulated in Table 6.1 for each indexing method.

Table 6.1: Average fingertip position error (Δ_p) and average viewpoint error (Δ_ϕ) compared with ground truth for nearest neighbour.

Method	Δ_p, cm	Δ_ϕ
Chamfer Distance (d_{SC})	4.34	30.7 °
Areas and Means (d^*)	3.81	26.5 °
Two-tier ($N = 32$)	3.30	24.0°
Two-tier ($N = 64$)	3.27	24.0°
Two-tier ($N = 128$)	3.22	23.7 °
Direct EMD (d)	3.25	23.9 °
Lower Bound	1.50	13.9 °

Table 6.2: Average fingertip position error per finger compared with ground truth for nearest neighbour.

Method	Δ_{Thumb}, cm	Δ_{Index}, cm	Δ_{Middle}, cm	Δ_{Ring}, cm	Δ_{Pinky}, cm
Chamfer Distance (d_{SC})	4.42	4.53	4.73	4.23	3.78
Areas and Means (d^*)	4.24	3.78	3.94	3.58	3.52
Two-tier ($N = 32$)	3.73	3.36	3.38	3.02	3.01
Two-tier ($N = 64$)	3.74	3.26	3.38	2.97	3.02
Two-tier ($N = 128$)	3.67	3.25	3.30	2.89	2.98
Direct EMD (d)	3.72	3.26	3.37	2.90	3.02
Lower Bound	2.01	1.49	1.63	1.36	1.03

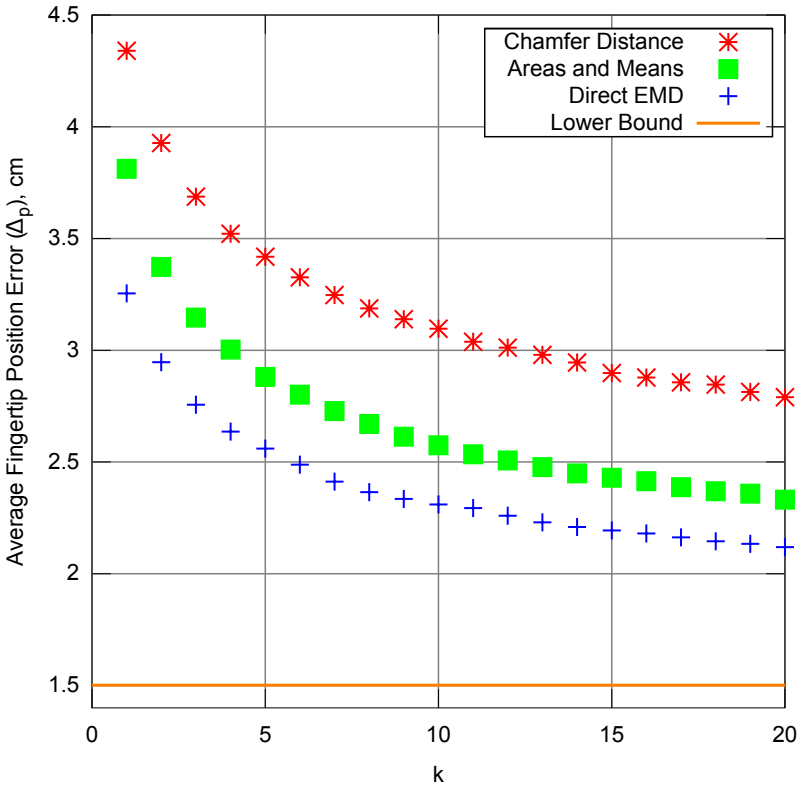


Figure 6.3: A comparison of idealised performance of multiple hypothesis tracking for: ($*$) The baseline system using the symmetric chamfer distance d_{SC}, based on [8]. (■) The metric which considers only the marker areas and means d^*. ($+$) Exact queries using the compound EMD d. Average fingertip position error (Δ_p) for the best match amongst k nearest neighbours is plotted for values of k ranging from 1 to 20. This plot demonstrates that the performance of d^* and d compares favourably with that of an existing system d_{SC}.

6.3 Discussion

6.3.1 Accuracy

To place the values in Table 6.1 and 6.2 into perspective, one may examine the average minimum quantisation error of fingertip position within the synthetic database. The minimum quantisation error for a finger setting is defined as the distance between the fingertip for the setting and the closest fingertip of a different finger setting. The average value over a finger's different settings is then the average minimum quantisation error.

A large male hand was used as the source of model dimensions. As the same hand was used to record the test images, the absolute scale of the hand model is known. Therefore, while the system can in general not determine the absolute size of the hand, in this case ground truth measurements are known, and can be used to quote errors in terms of, for example, centimetres.

The average minimum quantisation errors for the thumb, index, middle, ring and pinky fingers were 4.21 cm, 2.57 cm, 3.26 cm, 2.92 cm and 2.69 cm respectively. The average over all

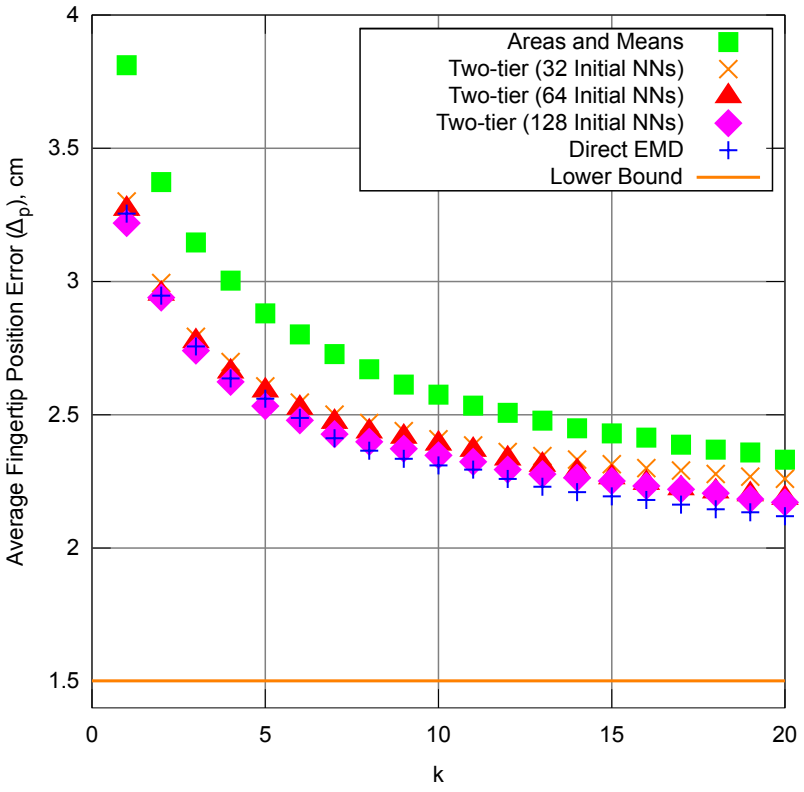


Figure 6.4: A comparison of idealised performance of multiple hypothesis tracking for: (■) The metric which considers only the marker areas and means d^*. (×,▲,♦) The two-tier approach which obtains an initial set of N nearest neighbours using d^*, followed by refinement using the compound EMD. Values of N considered were 32 (×), 64 (▲) and 128 (♦) respectively. (+) Exact queries using the compound EMD d. Average fingertip position error (Δ_p) for the best match amongst k nearest neighbours is plotted for values of k ranging from 1 to 20. This plot demonstrates the tendency of the two-tier approach's accuracy to match that of exact EMD queries more and more closely as N increases.

the fingers was 3.12 cm. Note that the individual values are roughly double the corresponding lower bounds on the individual average fingertip errors in Table 6.2.

It is clear from Figure 6.2 that direct application of the EMD returns plausible candidates as the closest match. The viewpoint and pose are well estimated in most of the images. This is borne out by the results from Table 6.1, showing that the fingertips are, on average, only 3.25 cm out of place. This value is approximately the same as the average minimum quantisation error averaged over all fingers (3.12 cm). This indicates that the hand shapes errors were about the same as the granularity due to the limited resolution of the database.

Estimation based solely on the areas and centroids on the markers performs surprisingly well. It can be observed that this method was able to achieve similar accuracy ($\Delta_\phi = 26.5°$) in determining the viewpoint when compared to direct application of the EMD ($\Delta_\phi = 23.9°$). However, the system is unable to respond to the detailed shape of each of the markers, and so estimation of the hand shape suffers. Direct application of the EMD outperforms this approach by 17%, based on the average fingertip error Δ_p. Note that taking the lower bound on the

average fingertip error as a baseline instead of zero may be more appropriate, in which case the performance increase is 32%. Table 6.2 reveals that the gains in accuracy between d^* and d are consistent across every individual finger, indicating a level of uniformity with respect to the modelling accuracy.

If one assumes that the correct match is returned within the set of closest neighbours, simple refinement of that set using the EMD should yield similar results to its global application. This hypothesis is supported by Figure 6.2 where, under close inspection, the results for the two-tier approach are often identical to those obtained by direct application of the EMD. If one takes into account that using this two-tiered search approach resulted in increasing the speed of the search by a factor of 160 (as discussed in Section 6.3.2), its value becomes apparent.

The closely matched performance of the two-tiered approach to direct application of the EMD is confirmed by the similarity of their measured errors in Table 6.1. As N is increased, a decrease in Δ_p is observed, approaching the value obtained using direct application of the EMD.

This trend continues when multiple hypotheses are admitted. In Figure 6.3, it can be seen that the pose estimation based on d^* (areas and means) and d (EMD) significantly outperform the baseline system d_{SC} (chamfer distance), with the EMD being the best performing dissimilarity measure. The effectiveness of the two-tier approach in approximating the performance of direct application of the EMD is illustrated by Figure 6.4. Here it can be easily observed that the two-tier search strategy approaches the performance of direct application of the EMD as N increases (the number of nearest neighbours obtained from the initial, rough search using marker areas and means). At $N = 128$, the two-tier approach exhibits virtually the same or better average fingertip position error for values of k up to around 7. The performance of the two-tier approach remains close to that of direct EMD application up to $k = 20$.

6.3.2 Runtime Behaviour

Runtime behaviour of the system was studied separately using a smaller, earlier dataset. Only exact nearest neighbour queries using VP-trees were considered. This was necessary in order for the accuracy of the two-tier approach to be studied in isolation from errors due to the use of approximate nearest neighbour queries on the VP-trees themselves. Because the chamfer distance is not a metric, exact search required brute force calculation for each pose in the database, and therefore no VP-tree runtime behaviour could be tested.

Weber et al. [81] observed that, when searching spaces with dimensionality in excess of about 10, performance of exact nearest neighbour queries using space partitioning strategies may become comparable to simple sequential search. Consistent with this observation, queries performed during the course of this study visited significant fractions of the database. In particular, in order to return the nearest neighbour ($k = 1$), the fraction of the database considered by the direct application of the EMD, the approach based only on areas and means and the two-tier approach ($N = 128$) were on average 16%, 13% and 22% respectively. Note that in the two-tier

approach, 128 initial nearest neighbour candidates were retrieved, and so more of the database needed to be visited.

Because significant fractions of the databases needed to be considered, query resolution was slow. The system was executed on an Intel Xeon (Nehalem) 2.8Ghz. Using direct application of the EMD, on average 2200s were needed to determine pose candidates. Simply using the areas and means reduced this time to 6.5s, while using the two-tier approach for refinement ($N = 128$) required 13s. Thus, the two-tier approach was approximately 160 times faster than using direct application of the EMD.

6.4 Summary

During the course of this chapter, based on an evaluation using 500 real test images, the following key findings were made.

It was found that both approximate and exact queries using compound EMD outperforms a closely related system, that of Athitsos et al. [8]. The two-tier search approximation was shown to approach the performance of the unapproximated compound EMD for relatively modest values of N (the number of initial nearest neighbours returned by the rough search). Because the rough metric is simple to evaluate, the two-tier approach vastly outperforms exact queries using the compound EMD by a factor of 160.

These findings demonstrate the utility of the two-tier approximation, and its applicability to hand pose estimation.

However, several additions need to be introduced to the system in order to provide the functionality that the tutor system requires. These modifications will be introduced in the next chapters. Note however that the fundamental aspects of the system's functioning remain unchanged, and therefore the results obtained in this chapter still hold.

Chapter 7

Automatic Colour Calibration

In this chapter, the determination of the colour models used to segment video images into candidate finger marker regions is considered. Obtaining colour models for coloured gloves is not simple. While the marker colours are roughly uniform, and the approximate hue of each marker is known, this prior knowledge is nonetheless still vague. The knowledge that the thumb marker is "red" does not indicate with enough certainty the exact shade of red present in the marker. Furthermore, this colour may vary significantly due to changes in illumination. Sign language processing systems use a variety of approaches to address these challenges.

Akmeliawati et al. [2] use manually defined thresholds in order to segment markers on a pair of gloves with four marker colours (yellow, blue, red and purple). As this approach does not allow easy adaptation to different lighting conditions, or variations in glove manufacture, it is not suitable for general application.

Burger et al. [20], Zieren [91] and Wang and Popović [80] use manually segmented calibration images to estimate colour models. In such cases, a set of pixels known to be part of a specific marker is available, and so estimating the marker colour model becomes relatively simple. However, manual segmentation is cumbersome, and cannot reasonably be expected of an end user.

Zhang et al. [90] forego a detailed colour model for a rough search of the hand region and then using a region growing technique which takes into account constraints on region shape. Because the resulting features are based on ellipses, the shape constraints are relatively simple to impose. This method is not applicable to the case of the compound Earth Mover's Distance (EMD), as contour information of high fidelity is needed, rather than simple ellipse features. Additionally, region growing techniques can be slow at runtime, whereas segmentation based on colour model lookup tables can be performed rapidly.

To avoid the brittleness associated with manually defined thresholds, the cumbersomeness of manually segmented calibration images, and the computational complexity of constrained region growing methods, an automatic colour calibration procedure will be presented in this chapter. The core of the calibration procedure is a graphical model, which is used to reason probabilistically about calibration images. A high-level overview of the calibration procedure

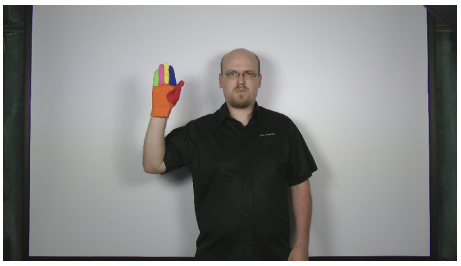


Figure 7.1: Example of a calibration image, showing the user assuming the pose required by the automatic colour calibration system.

will first be provided, followed by a more formal discussion of the graphical model.

The procedure is composed of the following stages:

- The user is asked to raise their right hand, with fingers together, palm facing towards the camera and pointing upward. This pose is illustrated in Figure 7.1.
- A hand detection procedure uses rough colour models for the markers in combination with the known pose to locate the hand within the image.
- The location and scale of the detected hand are used to establish a set of spatial priors for the location of marker pixels.
- The rough colour models, spatial priors and pixel colours are presented to a graphical model. The model reasons about this information by means of loopy belief propagation [11]. The process produces a set of refined colour models which may be used at runtime.

Colour models used during segmentation are defined using the HSV (hue, saturation and value) colour model. Hue and saturation may be used to capture the "shade" of a colour, while value represents its brightness. Figure 7.2 illustrates variation of hue and saturation with constant value (brightness). As value is heavily dependent on the intensity of incident illumination, trainable colour models employed during image segmentation are based on hue and saturation only (fixed priors over value are still employed during segmentation).

7.1 Hand detection

Initially, only a broad indication of the colour of each marker is available. This can be quantified by defining a set of rough colour models. Figure 7.3 shows the set of rough colour models used in the initial localisation of the hand, as well as the likelihood images generated by mapping the pixels of the calibration image using the rough colour models. The models are illustrated as hue-saturation planes such as in Figure 7.2, with the value component of each hue-saturation pair used to indicate the relative probability mass assigned to the pair. Brighter regions of the hue-saturation plane are more probable given the colour model.

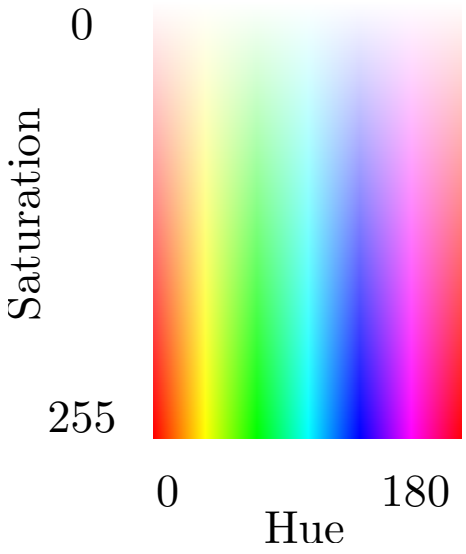


Figure 7.2: The hue-saturation plane with maximum value (brightness). Note that hue is periodic. The domains of the hue and saturation reflect their representation within the colour calibration system as unsigned bytes.

Note in Figure 7.3 that there is substantial initial confusion between similarly coloured markers. In addition, substantial non-glove regions are also assigned significant likelihoods, and so are potential distractors if the rough colour models are used directly for image segmentation.

At this stage, the system combines knowledge of the user's hand pose with the likelihood images generated using the rough colour models. In order to do this, an approach inspired by the Viola-Jones facial detector [72] is employed. Each marker is assigned one approximating rectangle (or two, in the case of the thumb) as illustrated in Figure 7.4a. These rectangles roughly correspond to the position of the markers when the user assumes the pose required by the automatic colour calibration, with the rectangles having predefined relative proportions.

This hand template can be easily scaled and translated anywhere in the image. The detection process works by considering many such scales σ and translations (x,y). For each case, the rectangles associated with each marker are overlayed on the corresponding likelihood image generated using the rough colour models (as illustrated in Figure 7.4b). All the likelihoods within each rectangular region are summed together, producing a number $\rho_m(x,y,\sigma)$ for each marker region. The candidate scale and translation is then labelled with the score

$$\lambda(x,y,\sigma) = \prod_{m=1}^{6} \frac{\rho_m(x,y,\sigma)}{\sigma^2}$$

where σ is the scale of the candidate, and (x,y) is the location of the centre of the template. The division by σ^2 ensures that scores of small templates are adjusted as needed, as area grows in proportion to the square of the scale. These scores are illustrated in Figure 7.4c.

The hand is then detected by finding the scale and translation which maximises λ:

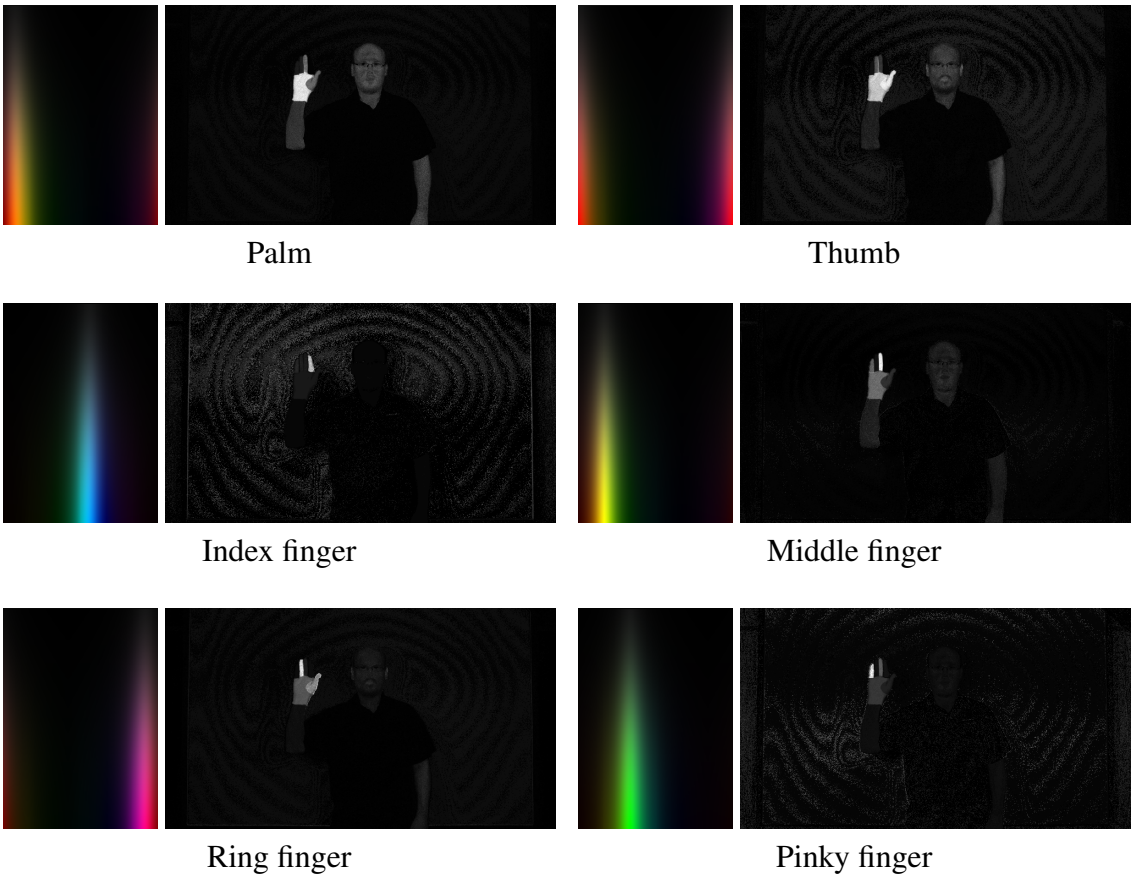


Figure 7.3: Rough prior colour models used in colour calibration, and the resulting likelihood images obtained from the calibration image in Figure 7.1.

$$(x, y, \sigma)_{\text{hand}} = \underset{(x,y,\sigma)}{\operatorname{argmax}}\, \lambda(x, y, \sigma).$$

This optimal candidate is shown in Figure 7.4e, which corresponds to the sharp peak marked with a red dot in Figure 7.4c at scale 45. Note the tendency of the detected maximum to be sharply peaked at the correct scale.

The scores plotted in Figure 7.4c are obtained from a subimage of the original calibration image, shown in Figure 7.4d. The lightest, centre block is the domain of (x, y). The larger block is the furthest extent of the hand templates (those at scale 70) that are centred at the edge of the inner block. The rest of the image is not considered in the example. The actual system operates on the entire image. The restricted region considered in Figure 7.4d was chosen so that the sharp peaking in λ at the maximum candidate is clearly visible.

Note that, as in [72], the summations inside the rectangles can be performed rapidly by first forming the integral image of each likelihood image

$$I(x, y) = \sum_{x'=0}^{x} \sum_{y'=0}^{y} i(x', y')$$

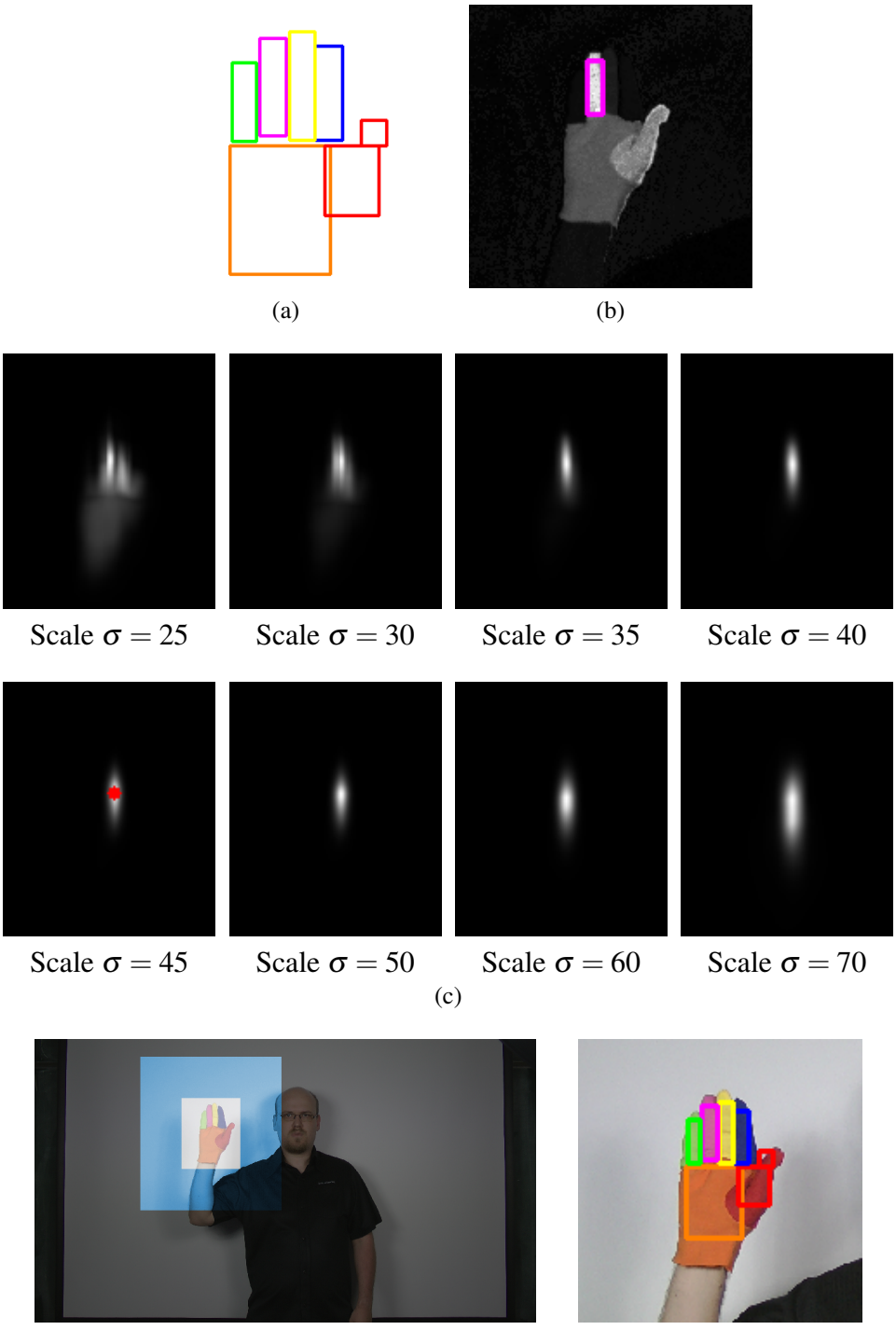


Figure 7.4: Locating the hand during colour calibration: (a) Rectangular areas corresponding to different marker regions. These boxes are superimposed on each marker's respective rough likelihood image. (b) The ring finger likelihood image with the box associated with the detected hand superimposed. ρ_m for this finger is the sum of all the likelihood values inside the box. (c) Values for $\lambda(x, y, \sigma)$ plotted at different scales. The images correspond to the lightest, inner block in (d), with the outer block indicating the maximum reach of any templates centred in the inner region (see main text for explanation). (e) The detected hand, corresponding to the maximum of $\lambda(x, y, \sigma)$ marked with a red dot in (c).

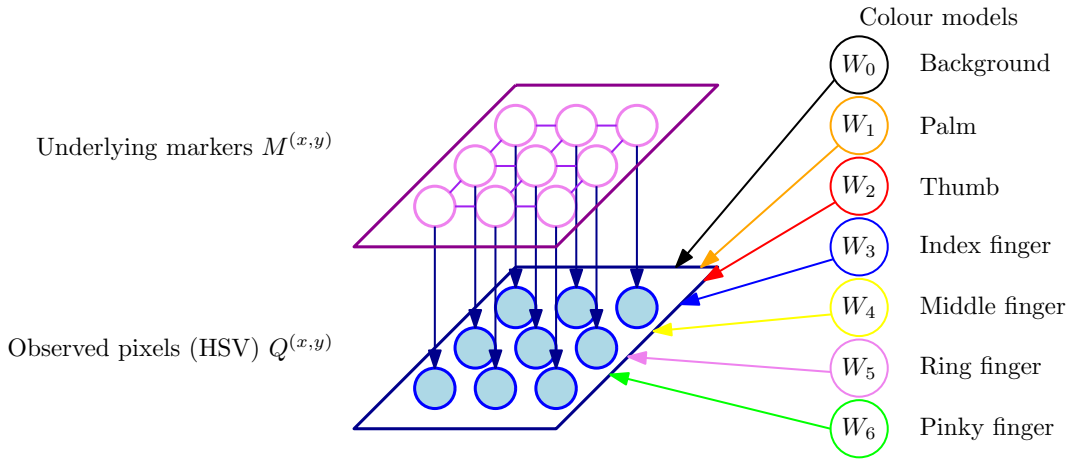


Figure 7.5: A simplified overview of the autocalibration graphical model. The colours of observed pixels (bottom layer) are determined by the marker underlying each pixel (top layer) and the colour model of the underlying markers on the right. These influences are represented by directed links. Neighbouring underlying markers influence each other, represented by undirected links inside the marker layer.

where $I(x,y)$ is the integral image, and $i(x,y)$ is the original likelihood image. A summation over any rectangular region with corners (x_0,y_0) and (x_1,y_1) (with $0 < x_0 \leq x_1$ and $0 < y_0 \leq y_1$) in $i(x,y)$ can then be evaluated using only four values from $I(x,y)$:

$$\sum_{x=x_0}^{x_1} \sum_{y=y_0}^{y_1} i(x,y) = I(x_1,y_1) - I(x_0-1,y_1) - I(x_1,y_0-1) + I(x_0-1,y_0-1).$$

Given the integral images, the values of $\lambda(x,y,\sigma)$ can therefore be calculated rapidly for many different possible (x,y,σ). Depending on the resolution of the input image, and the extent of the search region, the search process was found to take on the order of one second per frame. This is sufficiently rapid for practical application, because colour calibration is needed only once per tutor session.

In the final system, each frame of the colour calibration input video was processed as discussed, and the maximum $\lambda(x,y,\sigma)$ over all frames of the calibration video was chosen as the final candidate. This has the effect of choosing the best candidate from all the images available in the calibration video.

7.2 Graphical model overview

Using the location of the hand in the calibration image and prior information about the colour of the markers, it is possible to reason statistically in order to find a refined set of colour model posteriors. The autocalibration system makes use of a graphical model [11] for this inference task. A broad overview of the model's operation will be given in this section.

A simplified view of the model is provided in Figure 7.5. Inference performed directly on this simplified model may be intractable, and this issue is addressed in the subsequent sections.

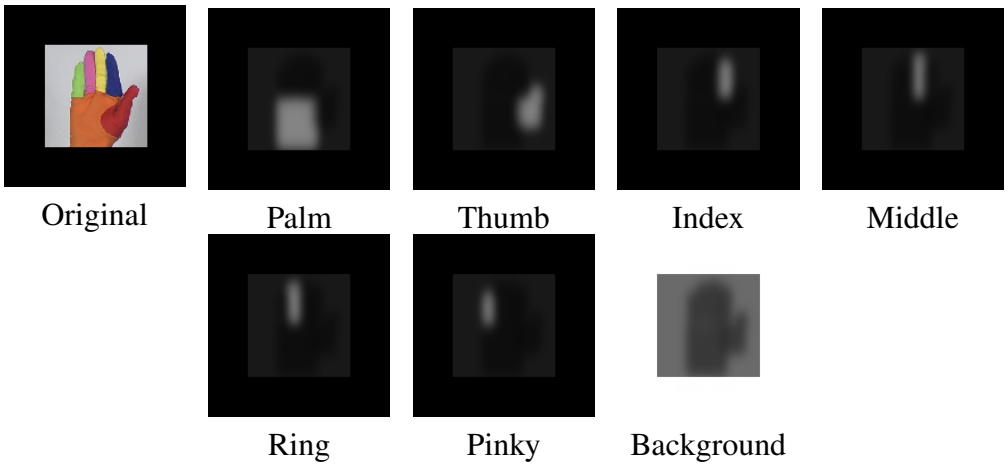


Figure 7.6: Spatial priors which are derived from the rectangular marker areas obtained from hand localisation. These priors represent initial beliefs about the locations of the pixels associated with each finger marker. The priors are refined during belief propagation.

The model in Figure 7.5 consists of three main subcomponents. The bottom layer consists of all the hues, saturations and values of each individual pixel in the region of the calibration image immediately around the detected hand. These are known a priori, and so are constants during the inference process.

It is assumed that each pixel $Q^{(x,y)}$ is caused by some underlying marker, with the background being a possible "marker". The variables representing the underlying marker for each pixel are contained in the top layer of the model, with each pixel in the bottom layer having a corresponding marker given by $M^{(x,y)}$ in the top layer. The influence of the underlying marker on the observed pixel is represented by a directed link from the marker to the pixel.

Knowledge of the underlying marker $M^{(x,y)}$ is not enough to determine pixel colour $Q^{(x,y)}$, as each marker must also have a colour model which, in conjunction with knowledge of *which* marker underlies the pixel, determines the colour of that pixel. The colour models W_m are depicted on the right. The directed links emanating from the colour models influence each of the pixels in the bottom layer.

The autocalibration system uses the maximum candidate obtained from hand localisation (as illustrated in Figure 7.4e) to create a set of *spatial priors* over the variables representing the underlying marker for each pixel $M^{(x,y)}$. The spatial priors are illustrated in Figure 7.6. Here, the probability associated with each marker at each pixel is illustrated. A lighter pixel means it is probable (according to the current state of knowledge) that this pixel is caused by the marker. The priors are generated by drawing filled rectangles corresponding to the template rectangles into each marker probability mask. The masks are smoothed using a Gaussian kernel, to allow for uncertainty in the marker locations. Finally, the potential at each pixel is normalised, so as to produce a valid probability distribution $p(M^{(x,y)})$ at each underlying marker.

Prior information about the colour models is also introduced. Figure 7.7 shows visualisations of these prior distributions. Note that these are similar, though not identical, to the priors

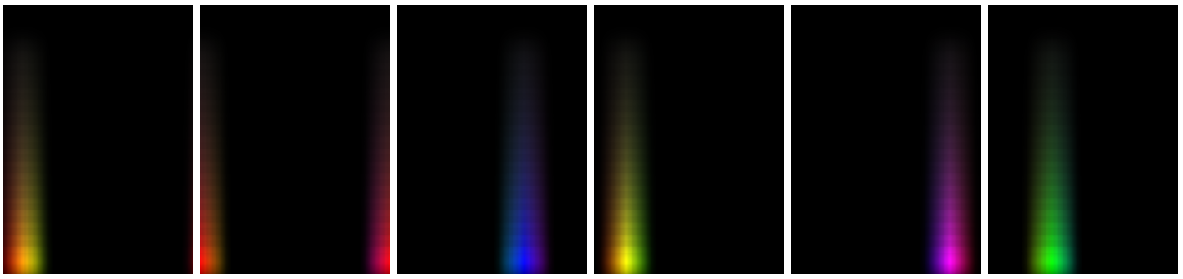


Figure 7.7: The prior colour models at the start of the inference process.

used in the hand localisation.

Lastly, spatial coherence is enforced by introducing constraints between neighbouring underlying markers, represented by the undirected links in the top layer. The top layer, therefore, forms a Markov Random Field [13]. These constraints make it more likely that neighbouring underlying markers are the same. This introduces a measure of noise resistance into the model.

Loopy belief propagation [11] is now used to perform inference. Initially, an upward pass is performed through the network. The pixel colours pass messages to the underlying markers and the colour models, essentially providing a first estimate of the colour models which combines with the prior colour models. This combined colour model is then enforced in a downward pass. This is followed by message passing over the undirected links in the underlying marker layer, which attempts to smooth the segmentation spatially. These message passing phases repeat until convergence is achieved.

In the following sections, a more detailed overview of each aspect of the graphical model and inference process is given.

7.3 Notation

Before further discussion, the notation [43] used to describe the mathematical aspects of the graphical model can be summarised is as follows:

R A random variable.

r An instantiation of random variable.

$\mathbf{R}$ Abbreviation for a set of random variables $R_1, \ldots, R_N$, where superscripts or subscripts are clear given the context.

$p(R_1, \ldots, R_N | Z_1, \ldots, Z_K)$ A probability distribution over the random variables $R_1, \ldots, R_N$, conditioned on the random variables $Z_1, \ldots, Z_K$. The random variables may assume any value in their domain.

$p(\mathbf{R}|\mathbf{Z})$ An abbreviation for the preceding probability distribution.

$p(R_1 = r_1, \ldots, R_N = r_N | Z_1 = z_1, \ldots, Z_K = z_K)$ A probability distribution over the random variables $R_1, \ldots, R_N$, conditioned on the random variables $Z_1, \ldots, Z_K$. The random variables are forced to assume the given values, therefore $R_1 = r_1$ means that R_1 is the constant r_1.

$p(\mathbf{R} = \mathbf{r} | \mathbf{Z} = \mathbf{z})$ An abbreviation for the preceding probability distribution, where $\mathbf{r}$ and $\mathbf{z}$ are the constant values of the random variables.

All variables have discrete and finite domains, as discussed during the course of the following sections.

7.4 Colour models

In the calibration system, the HSV colour model is used to represent colours. For the purposes of inference, colours are divided into two components. The first component is the ordered pair formed by a pixel's hue and saturation components, and is denoted by $C^{(x,y)}$.[1] Hue is modelled as an integer ranging from 0 to 180 (so as to fit inside a single byte), and saturation is modelled as an integer ranging from 0 to 255. However, the hue and saturation are quantised by setting the least significant bit of the hue to zero, and the three least significant bits of the saturation to zero. This limits the domain size of the hue-saturation pairs.

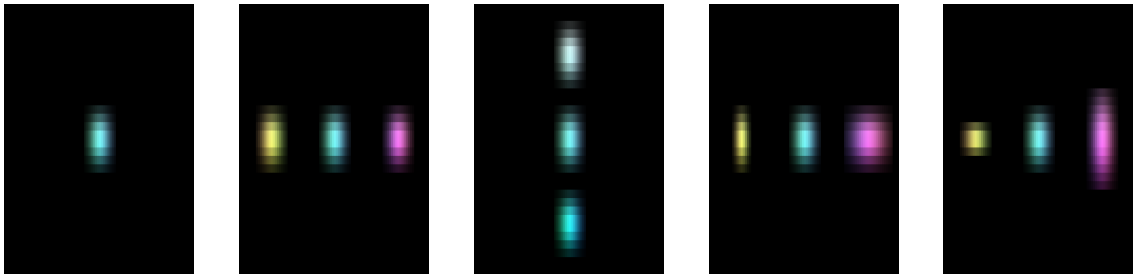
The second component of the pixel colour is its value, denoted $V^{(x,y)}$, and is modelled as an integer ranging from 0 to 255. Because the value component of a colour is strongly influenced by ambient illumination, it is does not form part of the colour model. A colour model for the marker m is then simply a probability distribution $p(C^* | M = m)$ which describes how probable each hue-saturation pair is, given that the marker is m. For the colour calibration system, m may assume any integer value from 0 to 6, indicating the background "marker", the palm, the thumb, the index finger, the middle finger, the ring finger and the pinky finger respectively.[2]

However, direct control over $p(C^* | M = m)$ may lead to overfitting of the colour models, as it is possible to control the probability of each possible hue-saturation pair individually. Instead, a set of component colour models $p(C^* | M = m, W_m)$ are defined. W_m has a number of possible settings which act as parameters for the component colour models. A distribution over these settings $p(W_m | \mathbf{C} = \mathbf{c}, \mathbf{V} = \mathbf{v})$ is now the underlying cause for a given colour where $\mathbf{c}$ and $\mathbf{v}$ are the hue-saturation pairs and values of the calibration image. The final colour model (as used during marker segmentation) is then obtained by marginalisation

$$p(C^* | M = m, \mathbf{C} = \mathbf{c}, \mathbf{V} = \mathbf{v}) = \sum_{W_m} p(C^* | M = m, W_m = w_m) p(W_m = w_m | \mathbf{C} = \mathbf{c}, \mathbf{V} = \mathbf{v}).$$

[1] Note that the superscript (x,y) indicates the quantity is associated with the pixel at the position (x,y).

[2] Note that C^* represents any pixel during run-time segmentation of markers, and *not* a pixel considered during colour calibration $C^{(x,y)}$. The calibration-time variables $C^{(x,y)}$ are collected in $\mathbf{C}$, which *excludes* C^* (a run-time variable).



(a) (b) (c) (d) (e)

Figure 7.8: Colour models and the effect of their parameters: (a) A single colour model with $\mu^H_{w_m} = 90.0$, $\mu^S_{w_m} = 128.0$, $\sigma^H_{w_m} = 7.0$, and $\sigma^S_{w_m} = 15.0$. Subsequent examples are the superposition of three different colour models in each case. Note that the anisotropic pixelation is due to the different levels of quantisation of the hue and the saturation. (b) The effect of changing the hue's mean $\mu^H_{w_m}$. The leftmost model has $\mu^H_{w_m} = 30.0$, the middle $\mu^H_{w_m} = 90.0$ and the rightmost has $\mu^H_{w_m} = 150.0$. (c) The effect of changing the saturation's mean $\mu^S_{w_m}$. The topmost model has $\mu^S_{w_m} = 48.0$, the middle $\mu^S_{w_m} = 128.0$ and the bottom model has $\mu^S_{w_m} = 208.0$. (d) The effect of changing the hue's standard deviation $\sigma^H_{w_m}$. The leftmost model has $\sigma^H_{w_m} = 4.1$, the middle $\sigma^H_{w_m} = 7.0$ and the rightmost has $\sigma^H_{w_m} = 11.9$. (d) The effect of changing the saturation's standard deviation $\sigma^S_{w_m}$. The leftmost model has $\sigma^S_{w_m} = 8.8$, the middle $\sigma^H_{w_m} = 15.0$ and the rightmost has $\sigma^S_{w_m} = 25.5$.

Therefore, the probability of a hue-saturation pair C^*, given that the marker is m, is the sum of all the probabilities of C^* for each of the component colour models scaled by the probability inferred for that component model $p(W_m = w_m|\mathbf{C},\mathbf{V})$.

The component colour models are defined as truncated two-dimensional discrete-valued pseudo-Gaussians over the hue h and the saturation s of the hue-saturation pair $c = (h,s)$. The pseudo-Gaussians referred to are discrete-valued distributions that derive their probabilities from a Gaussian envelope, but sampled, and followed by normalisation. The pseudo-Gaussians have four parameters. If the component colour model is w_m, the mean of the hue is $\mu^H_{w_m}$ and its standard deviation is $\sigma^H_{w_m}$. The mean of the saturation is $\mu^S_{w_m}$ and its standard deviation is $\sigma^S_{w_m}$. The effects of these different parameters are illustrated in Figure 7.8. The general form of the component colour model is then

$$p(C = c = (h,s)|M = m, W_m = w_m) \propto \pi(h,s,w_m) \exp\left[\frac{-d_H^2(h,\mu^H_{w_m})}{(\sigma^H_{w_m})^2}\right] \exp\left[\frac{-|s-\mu^S_{w_m}|^2}{(\sigma^S_{w_m})^2}\right] \tag{7.4.1}$$

where d_H is the hue distance and $\pi(h,s,w_m)$ is a windowing function which truncates the distribution. The hue distance d_H is used because of the periodicity of hue, and so a simple absolute difference between the hue and its mean would not work as intended. Assume that all hues range from 0 to H_{max}, then define the hue distance as

$$d_H(h_1,h_2) = \min(|h_1 - h_2|, |[H_{max} + \min(h_1,h_2)] - \max(h_1,h_2)|)$$

which is the distance between h_1 and h_2, either directly or via a jump between 0 and H_{max}.

Marker	Variable	$\mu^H_{w_m}$			$\mu^S_{w_m}$			$\sigma^H_{w_m}$	$\sigma^S_{w_m}$
		Min	Max	Divisions	Min	Max	Divisions		
Background	W_0	0	180	40	0	250	21	5	10
Palm	W_1	0	36	8	40	250	18	5	10
Thumb	W_2	162	18	8	40	250	18	5	10
Index finger	W_3	102	138	8	40	250	18	5	10
Middle finger	W_4	12	48	8	40	250	18	5	10
Ring finger	W_5	132	168	8	40	250	18	5	10
Pinky finger	W_6	42	78	8	40	250	18	5	10

Table 7.1: Table of colour model parameters. The parameters $\mu^H_{w_m}$ and $\mu^S_{w_m}$ may take on values in the evenly discretised range from 'Min' and 'Max' with the indicated number of 'Divisions'. Note in the case of the thumb, 'Min' may be larger than 'Max' for $\mu^H_{w_m}$, because hue is periodic.

The windowing function $\pi(h,s,w_m)$ truncates the distribution so that the probabilities for any hues h or saturations s beyond two standard deviations $\sigma^H_{w_m}$ or $\sigma^S_{w_m}$ are driven to zero. The windowing function is defined more formally as

$$\pi(h,s,w_m) = \begin{cases} 0 & d_H(h,\mu^H_{w_m}) > 2\sigma^H_{w_m} \text{ or } |s-\mu^S_{w_m}| > 2\sigma^S_{w_m} \\ 1 & \text{otherwise} \end{cases} .$$

Each marker m has associated with it a number of possible values for the component colour models w_m. In the current system, the component colour models of marker m have constant standard deviations for hue $\sigma^H_{w_m}$ and saturation $\sigma^S_{w_m}$. The means $\mu^H_{w_m}$ and $\mu^S_{w_m}$ are evenly spaced on a grid, with the centre hue of the grid being different for each marker. The hue is restricted within a range of allowable values, roughly corresponding to the color associated with each glove marker. The background model is a special case. It shares the same standard deviations of the other models, but the means are on a regular grid that spans all possible hues and saturations. This is because the background can contain objects of any colour, and so the model must reflect this.

The domain of each marker's colour model parameters is recorded in Table 7.1. The domain of each color model parameter W_m is then the Cartesian product of the domains of $\mu^H_{w_m}$, $\mu^S_{w_m}$, $\sigma^H_{w_m}$ and $\sigma^S_{w_m}$.

7.5 Cluster graphs

Cluster graphs [43] are a formalism by which graphical models may be defined, and which allows flexible specification of the distributed representation of knowledge regarding each variable within the network, and how this is communicated amongst nodes. This formalism will be used to specify the graphical model used for inferring information about the colour models.

Graphical models provide a method for reasoning about probability distributions that can be factorised in a way which eases subsequent inference. More generally, the distributions need

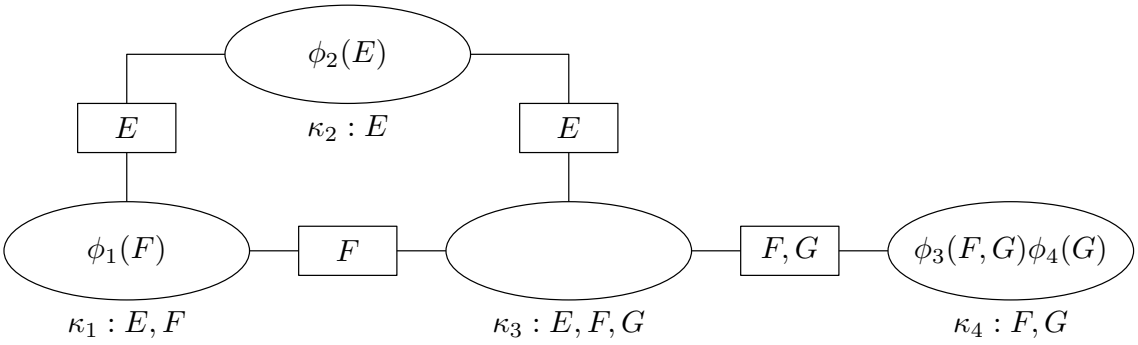


Figure 7.9: Example of a cluster graph. Clusters are indicated by ellipses which contain the potentials assigned to them. Each connection between two clusters has a rectangular sepset which contains the sepset variables. For purposes of illustration, each cluster is labelled κ_n : followed by the cluster variables. In practice, however, cluster variables are normally left implicit, and are inferred from a cluster's assigned potentials and connecting sepsets.

not be normalised, in which case they are referred to as potentials. Consider the following example of the potential ϕ which is known to be factorisable as follows:

$$\phi(E,F,G) = \phi_1(F)\phi_2(E)\phi_3(F,G)\phi_4(G). \tag{7.5.1}$$

Each of the potentials ϕ_n represent local relationships amongst the variables on which the potential depends. High values of $\phi_3(F,G)$ indicate combinations of F and G which are, when considering only ϕ_3, energetically (and hence probabilistically) favourable. However, these local relationships interact with each other by virtue of shared variables. For example, even though $\phi_1(F = f_0)$ may have a high value, f_0 will not be a possible instantiation of F if $\phi_3(F = f_0, G) = 0$.

Cluster graphs provide a means of representing the interactions among variables, so that posterior information may be inferred about them. A cluster graph consists of two component types, clusters and sepsets. These are illustrated in Figure 7.9, which shows a possible cluster graph for the decomposition in Equation 7.5.1.

Elliptical nodes are referred to as clusters, which have associated with them a set of random variables, referred to as their cluster variables. In the figure, there are four clusters, κ_1 through κ_4. The cluster variables are listed explicitly in this example after the colon in the labels $\kappa_n : \ldots$. Clusters act as repositories for information about the variables assigned to them, though this information may be distributed across a number of clusters.

In message passing algorithms, clusters communicate with each other by means of the edges between them, forming a consensus[3] regarding the probability assigned to different instantiations of the random variables associated with them. On the edges between clusters are rectangu-

[3]Standard belief propagation is only guaranteed to converge to the correct solution in polytrees. In graphs containing cycles, such as the graph used to infer the colour models, "loopy" belief propagation may still be performed [43], but there is no guarantee of convergence. If convergence occurs, there is no guarantee that the inferred distribution is correct. However, in practice, loopy belief propagation often produces useful results, as judged by empirical analysis.

lar nodes labelled with a subset of the intersection of the variables in the clusters connected by the edge. This label is called a sepset. Information communicated over an edge during message passing pertains only to those variables in the sepset. In the figure, the sepsets between κ_1 and κ_2, κ_2 and κ_3, and κ_3 and κ_4 are exactly the intersection of the variable sets of the clusters they connect.

The sepset $\{F\}$ between κ_1 and κ_3 is, however, a subset of the intersection of the variables associated with κ_1 and κ_3 ($\{E,F\}$ and $\{E,F,G\}$ respectively), the full intersection being $\{E,F\} \cap \{E,F,G\} = \{E,F\}$. Assigning $\{E,F,G\}$ as the sepset between κ_1 and κ_3 would have been illegal, because G is not in the variable list of cluster κ_1.

Without further information, the resulting probability distribution over all variables is uniform (assuming discrete and finite random variable domains). Information can be introduced into a cluster graph in two ways. Firstly, any observed variables may be set to their observed values. Secondly, the information contained in the probability distributions (or potential functions in the more general case) must be introduced into the network by assigning each potential to a cluster. In the figure, there are four potentials: $\phi_1(F)$, $\phi_2(E)$, $\phi_3(F,G)$ and $\phi_4(G)$, which are assigned to κ_1, κ_2, κ_4 and κ_4 respectively. Potentials may be assigned only once. A potential may be assigned to a cluster only if the potential scope[4] is a subset of the cluster variables. Assigning $\phi_2(E)$ to κ_4 would be illegal because E is not in κ_4's set of variables. Clusters may have any number of potentials, or no potentials. So, for example, κ_3 has no potentials, κ_1 and κ_2 have one each, and κ_4 has two.

Cluster graphs must also exhibit the running intersection property[43]. This property holds if, for any two clusters κ_i and κ_j which both contain the variable X, there is one and only one path between the clusters such that X is in the sepsets of each edge along the path.

This implies three things:

- All nodes along the path must contain the variable X, otherwise the sepsets could not legally contain X. For example, the path from κ_1 to κ_4 in the Figure 7.9 contains F in every node and sepset. If F were missing from κ_3, then the sepsets of edges $\kappa_1 - \kappa_3$ and $\kappa_3 - \kappa_4$ could not legally contain F, violating the running intersection property.

- The edges and clusters that contain X form a polytree, meaning there are no cycles in the subgraph containing information about X. Such cycles would lead to "circular arguments" during message passing where poor information about the state of a variable is reinforced by the same poor information. In Figure 7.9, the nodes and edges containing E form a list, a very simple polytree. However, if the sepset of the edge $\kappa_1 - \kappa_3$ contained E, then the subgraph containing E would contain a cycle, violating this restriction.

- There are no isolated regions containing X. If this were the case, then in effect a set of random variables $X_1, \ldots, X_n$ have been introduced which partition the potentials associated with X between themselves based on where the potentials are assigned in the graph.

[4] The scope of a potential is the set of variables it depends on.

This means that each of these isolated variables are free to assume different values, which is inconsistent with the original probabilistic model. In the example, if the sepset of edge $\kappa_3 - \kappa_4$ did not contain F, then there would be two isolated subgraphs containing F. In particular, there is no path between cluster κ_4 and κ_1 containing F along all edges, despite both clusters containing F.

Apart from these restrictions, cluster graphs may be designed freely. The following sections present the components of the cluster graph used to perform inference in the colour calibration model. Firstly the model variables are enumerated, followed by the factorisation of the joint probability distribution. The nature of each of these factors is then discussed. The cluster graph is then discussed, as well as the message passing procedure employed. Finally, results obtained from experimental work are presented, followed by conclusions drawn from these results.

7.6 Model variables

A list of the random variables involved in the colour calibration procedure, and a description of each, follows:

$C^{(x,y)}$ The hue and saturation of the pixel at (x,y). This is always a constant given by the input image pixels. All $C^{(x,y)}$ in the model are collected in $\mathbf{C}$.

$V^{(x,y)}$ The value (brightness) of the pixel at (x,y). Value is modelled as an integer ranging from 0 to 255. This is always a constant given by the input image pixels. All $V^{(x,y)}$ in the model are collected in $\mathbf{V}$.

$M^{(x,y)}$ The underlying marker for the pixel at (x,y). It may assume any value between 0 and 6, indicating the background "marker", the palm, the thumb, the index finger, the middle finger, the ring finger and the pinky finger respectively. All $M^{(x,y)}$ in the model are collected in $\mathbf{M}$.

W_m Each marker's colour model is broken up into a set of smaller component colour models, where the final colour model takes the form of a posterior over all of these smaller models. The seven variables W_m each select one of the component colour models for the markers 0 through 6. Note that markers have their own set of component colour models, which is why there are seven W_m's. All W_m in the model are collected in $\mathbf{W}$.

7.7 Joint distribution factorisation

In this section, the assumed factorisation of the joint distribution over the model variables is described. The resulting conditional independences are illustrated in Figure 7.10, which represents this factorisation using a chain graph[11]. The joint probability distribution describing the

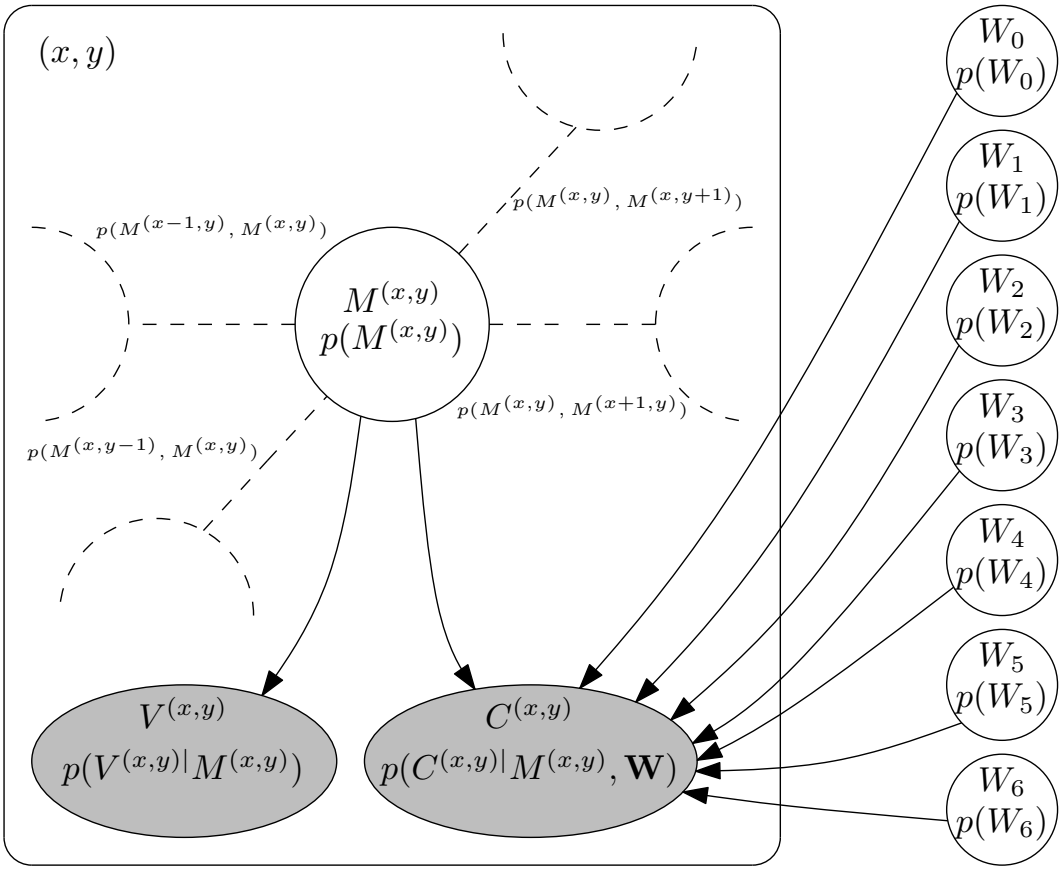


Figure 7.10: Chain graph capturing conditional independencies within the colour calibration model.

entire system is, by definition:

$$p(\mathbf{C}, \mathbf{V}, \mathbf{M}, \mathbf{W}). \tag{7.7.1}$$

A natural assumption would be that the hue, saturation and value of the pixels are determined by the underlying markers and their component colour models. From the definition of conditional probability

$$p(\mathbf{C}, \mathbf{V}, \mathbf{M}, \mathbf{W}) = p(\mathbf{C}, \mathbf{V}|\mathbf{M}, \mathbf{W})p(\mathbf{M}, \mathbf{W}). \tag{7.7.2}$$

It is assumed that the *priors* over the underlying markers and their component colour models are independent, leading to the factorisation

$$p(\mathbf{C}, \mathbf{V}|\mathbf{M}, \mathbf{W})p(\mathbf{M})p(\mathbf{W}). \tag{7.7.3}$$

When given the value of the underlying marker $M^{(x,y)}$ at pixel (x, y), and the currently active component colour model W_m, it is assumed that the pixel colours (hue, saturation and value) of individual pixels are independent. So the leftmost factor becomes

$$p(\mathbf{C}, \mathbf{V}|\mathbf{M}, \mathbf{W}) = \prod_{(x,y)} p(C^{(x,y)}, V^{(x,y)}|M^{(x,y)}, \mathbf{W}). \tag{7.7.4}$$

At each pixel, it is assumed that the colour models only describe the hue and saturation, that the hue-saturation values are independent of the value given the model and component models, and that the value is only dependent on the current marker. The per pixel factors now become

$$p(C^{(x,y)}, V^{(x,y)} | M^{(x,y)}, \mathbf{W}) = p(C^{(x,y)} | M^{(x,y)}, \mathbf{W}) p(V^{(x,y)} | M^{(x,y)}). \tag{7.7.5}$$

The factor $p(C^{(x,y)} | M^{(x,y)}, \mathbf{W})$ describes the probability of a hue-saturation pair given a specific marker, and a choice for all the component colour models of every marker. It is, however, dependent on all $\mathbf{W}$, making the dimensionality of the resulting potential too high to instantiate in practice. Because the value for $M^{(x,y)}$ determines which set of component colour models is active, the following factorisation is possible

$$p(C^{(x,y)} | M^{(x,y)}, \mathbf{W}) = \prod_{k=0}^{6} \phi_k(C^{(x,y)} | M^{(x,y)}, W_k) \tag{7.7.6}$$

where $\phi_k(C^{(x,y)} | M^{(x,y)}, W_k)$ assumes the role of individual marker colour model as discussed in Section 7.4. To ensure normalisation, $\phi_k(C^{(x,y)} | M^{(x,y)}, W_k)$ is set to one for $M^{(x,y)} \neq k$ (the use of ϕ versus p is used to indicate that the factorisation results in a set of unnormalised functions). When $M^{(x,y)} = k$, the factor $\phi_k(C^{(x,y)} | M^{(x,y)}, W_k)$ defines the component colour models for the marker k, with W_k acting as a switch between these colour models. This independence assumption has the advantage that a different set of colour models can be associated with each marker, while mitigating the cost associated with marginalisation over the added colour model variables. It has the disadvantage that $\phi_k(C^{(x,y)} | M^{(x,y)}, W_k)$ must be set to one for all the irrelevant markers. This means that, either memory is wasted by storing the additional entries, or an ad hoc special case must be introduced into the inference algorithm. Alternatively, one could use a single component model variable W, such that all markers share the same set of models, but this forces one to consider many irrelevant component colour models for each marker.

The distribution of the value component $p(V^{(x,y)} | M^{(x,y)})$ is assumed to be fixed, favouring higher values for hand markers, or exhibiting no preference in case of the background marker. If the value range is $0 \ldots 255$, the value distribution for hand markers is defined as the truncated sigmoid

$$p(V = v | M = m, m \neq 0) \propto \begin{cases} \frac{1}{1+\exp\left(-\frac{v-30}{15}\right)} & v \geq 20 \\ 0 & v < 20 \end{cases} \tag{7.7.7}$$

If M is the background marker, $p(V = v | m = 0) \propto 1$.

The factor $p(\mathbf{M})$ in Equation 7.7.3 describing the mutual interaction of markers underlying the pixels is defined as in a Markov Random Field (MRF) [11], such that only underlying marker variables from neighbouring pixels interact. These interactions are governed by joint probabilities $p(M^{(x-1,y)}, M^{(x,y)})$ of horizontally neighbouring pixel markers and $p(M^{(x,y-1)}, M^{(x,y)})$ of vertically neighbouring pixel markers. Furthermore, a spatial prior $p(M^{(x,y)})$ is introduced on each marker, which describes prior knowledge of where each finger is, roughly, in the image. This potential is given by

$$p(\mathbf{M}) = \frac{1}{Z_{\mathbf{M}}} \left[\prod_{(x,y)} p(M^{(x,y)}) \right] \left[\prod_{\substack{(x,y) \\ x>1}} p(M^{(x-1,y)}, M^{(x,y)}) \right] \left[\prod_{\substack{(x,y) \\ y>1}} p(M^{(x,y-1)}, M^{(x,y)}) \right]. \quad (7.7.8)$$

Here $\frac{1}{Z_{\mathbf{M}}}$ is the partition function which ensures $p(\mathbf{M})$ is normalised. The factors $p(M^{(x,y)})$ are obtained from the rough localisation of the hand and represent prior knowledge of where each marker might be in the calibration image. Examples of such spatial priors were illustrated in Figure 7.6. The neighbouring pixel constraints are proportional to

$$p(M = m, M' = m') \propto \begin{cases} 1.0 & m = m' \\ 0.3 & m \neq m' \end{cases}. \quad (7.7.9)$$

This potential favours configurations where neighbouring pixels share the same underlying marker, which acts to suppress noise in the resulting segmentation.

The factor $p(\mathbf{W})$ in Equation 7.7.3 represents the prior information available regarding the colour models. Assuming that the priors of the models are independent

$$p(\mathbf{W}) = \prod_{m=0}^{6} p(W_m). \quad (7.7.10)$$

Because the domains of W_m originate from the Cartesian product of the parameter domains of $\mu^H_{w_m}$, $\mu^S_{w_m}$, $\sigma^H_{w_m}$ and $\sigma^S_{w_m}$, as related in Table 7.1, $p(W_m)$ may be rewritten as a prior over these parameters of the form $p(\mu^H_{w_m}, \mu^S_{w_m}, \sigma^H_{w_m}, \sigma^S_{w_m})$. Because there is only one element in the domains of $\sigma^H_{w_m}$ and $\sigma^S_{w_m}$, this prior reduces to one of the form $p(\mu^H_{w_m}, \mu^S_{w_m})$, which is defined using descending exponentials as

$$p(\mu^H_{w_m}, \mu^S_{w_m}) \propto \exp\left[-\frac{1}{18} d_H(\mu^H_{w_m}, h^*_{w_m}) \right] \exp\left[-3 \left(1 - \frac{\mu^S_{w_m}}{256} \right) \right]. \quad (7.7.11)$$

Here d_H is the hue distance, $h^*_{w_m}$ is the "hue centre", which is the centre of the interval on which $\mu^H_{w_m}$ is defined, as shown in Table 7.1. This potential favours hues towards the centre of the range of $\mu^H_{w_m}$, and also more saturated colours.

The priors over the hue-saturation pairs may be obtained by performing the marginalisation

$$p(C|M = m) = \sum_{\mu^H_{w_m}} \sum_{\mu^S_{w_m}} p(C|M = m, \mu^H_{w_m}, \mu^S_{w_m}) p(\mu^H_{w_m}, \mu^S_{w_m}) \quad (7.7.12)$$

for each marker m. The results of this marginalisation were shown in Figure 7.7. The figure illustrates the preference for hues towards the centre of each range, and high saturations.

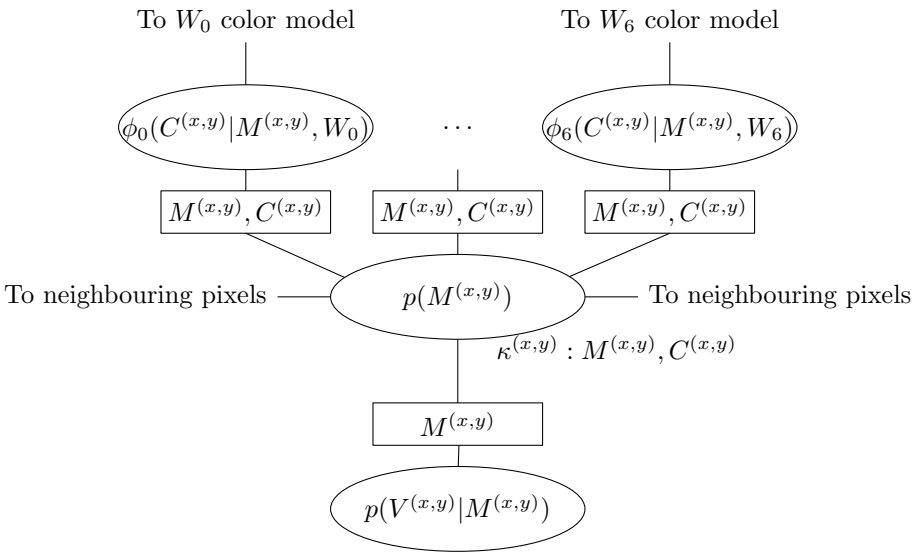


Figure 7.11: Clusters modelling each pixel in the calibration image.

7.8 Model potentials

In the previous section, the factorisation of the model's joint probability distribution was discussed, as well as the form of each of the factors. The factors of a model, in general, are referred as potentials. The potentials associated with the graphical model are now summarised:

$p(M^{(x,y)},M^{(x+1,y)})$**:** Horizontal marker compatibility potential. This favours horizontally neighbouring pixels which have the same underlying marker.

$p(M^{(x,y)},M^{(x,y+1)})$**:** Vertical marker compatibility potential. This favours vertically neighbouring pixels which have the same underlying marker.

$p(M^{(x,y)})$ Spatial prior derived from the location of the hand during calibration, which gives a rough indication of where the markers are in the calibration image.

$p(V^{(x,y)}|M^{(x,y)})$**:** Prior on pixel value (intensity) given the model underlying that pixel. This favours high values for glove markers, but shows no preference in the case of the background marker.

$\phi_m(C^{(x,y)}|M^{(x,y)},W_m)$**:** Potentials governing the relationship between colour model parameters and the distributions of hue-saturation pairs they represent.

$p(W_m)$**:** Potential encapsulating prior knowledge of the glove marker colour models.

7.9 Calibration cluster graph

The first portion of the cluster graph employed to perform inference in the colour calibration system, pertains to individual pixels and their associated underlying markers. This is illustrated

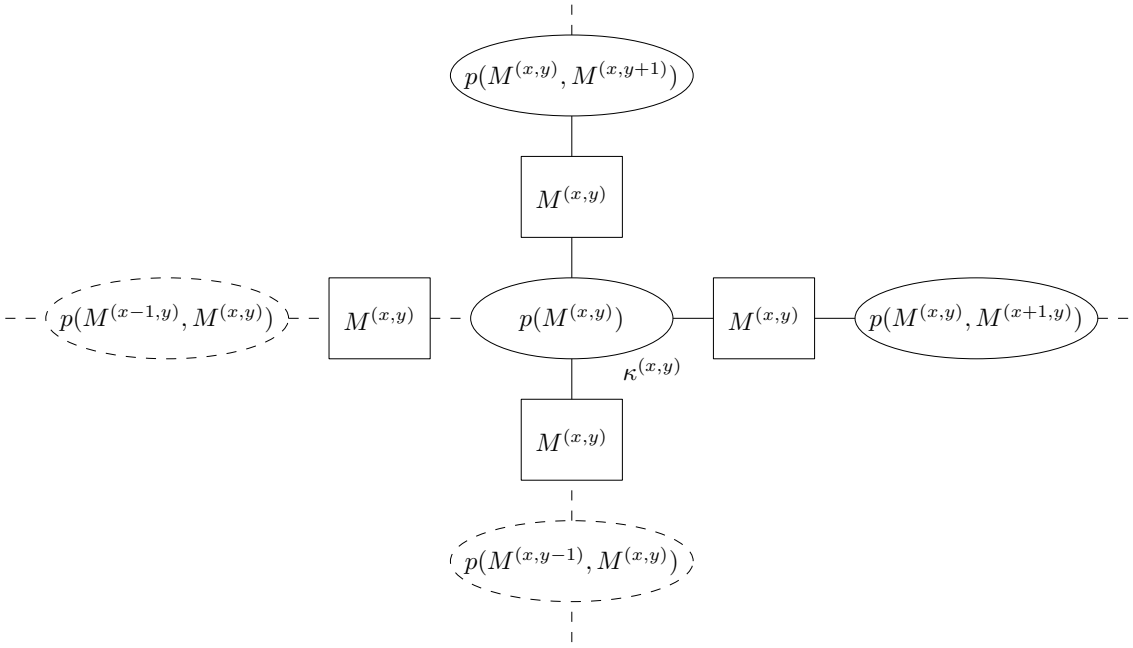


Figure 7.12: Clusters modelling constraints between neighbouring pixels.

in Figure 7.11. Note that $C^{(x,y)}$ and $V^{(x,y)}$ are constants, and so do not increase the dimensionality at the nodes where they appear.

The bottom cluster $M^{(x,y)}, V^{(x,y)}$ contains information about the relationship between underlying markers and the value at their pixels. Specifically, the priors $p(V^{(x,y)}|M^{(x,y)})$ from Equation 7.7.7 are attached to them. At the start of inference, a message is passed from these clusters upwards, weighting the marker beliefs in the central cluster according to the value at the pixel. The bottom cluster has no further effect on inference, as the message sent upwards will not change due to further iterations[5], and the downward message is unimportant for the purposes retrieving the colour models.

The central cluster $M^{(x,y)}, C^{(x,y)}$ contains information about the relationship between underlying markers and the hue and saturation at their pixels. The cluster aggregates information obtained from neighbour markers, and all the colour models. The potential $p(M^{(x,y)})$ represents one pixel in the spatial priors illustrated in Figure 7.6.

The upper clusters $M^{(x,y)}, C^{(x,y)}, W_m$ contain information relating the colour models W_m with the colour of the individual pixels $C^{(x,y)}$, based on current beliefs about the underlying marker at the pixel $M^{(x,y)}$. The potential $\phi_m(C^{(x,y)}|M^{(x,y)}, W_m)$, from Equation 7.7.6, enforces the component colour model with parameters W_m on the pixel colour $C^{(x,y)}$, given that $M^{(x,y)} = m$. Note again at this stage that $C^{(x,y)}$ is constant, and so it is the beliefs surrounding $M^{(x,y)}$ and W_m that mutually influence each other.

In Figure 7.12, mutual constraints between neighbouring pixels are represented. The cen-

[5]This is a general property of standard belief propagation. If a graph is divided into two subgraphs unconnected except for a single edge, the message sent from one subgraph to the other via the edge can be determined exactly if the originating subgraph is a polytree. In this particular case, the originating subgraph is a single node, a degenerate example of a polytree.

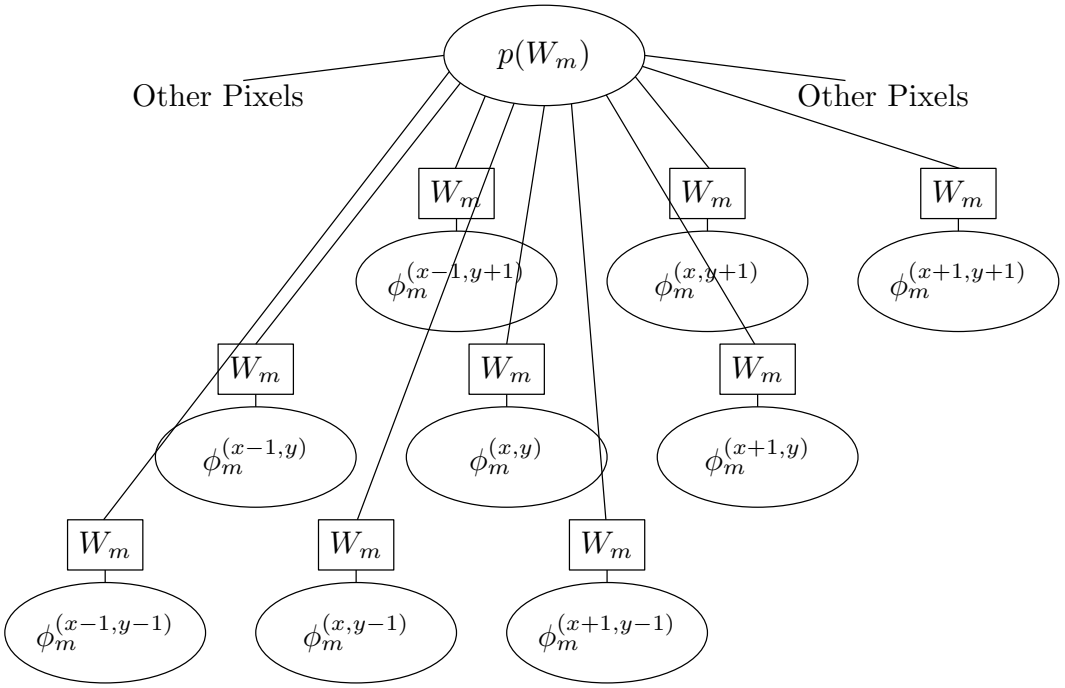


Figure 7.13: Idealised colour model cluster connections to pixels ($\phi_m^{(x,y)} = \phi_m(C^{(x,y)}|M^{(x,y)}, W_m)$).

tral cluster represents the same central cluster in Figure 7.11. The connecting clusters mediate information between this cluster and its counterparts within neighbouring pixels, and enforce constraints of the form $P(M, M')$, as in Equation 7.7.9. These potentials encourage neighbouring markers to be the same, which acts to suppress segmentation noise.

The information available at the top of Figure 7.11 regarding the colour model beliefs at each pixel must be aggregated for each colour model separately. In idealised form, this is illustrated in Figure 7.13. The prior $p(W_m)$ over the component colour models is associated with the top cluster. However, this form is intractable during inference. This is because each message from the top cluster to one of the lower clusters needs to take into account each of the other connections, an $O(N^2)$ operation when the resolution of the image is $N \times N$. As this has to be repeated for each pixel, the total operation is $O(N^4)$.

Two potential solutions include choosing each downward message as being approximately equal, reducing the complexity to $O(N^2)$. This was not tested, as it requires ad hoc alterations to the inference process. Another approach, which was used in the system, is to note that message calculations share partial results in a manner which admits an efficient decomposition of the process.

The single colour model cluster at the top of Figure 7.13 can be decomposed into a quadtree of identical clusters W_m with sepsets W_m. The apex of the tree, in the centre of the figure, is associated with the prior $p(W_m)$, and is also the cluster where the posterior colour model is calculated after convergence. Note that, at each node, the messages along edges only need to take into account at most four other messages, and so the operation at each node is $O(1)$. If $N^2 = 4^K$, then the number of nodes in the quadtree is $\sum_{k=0}^{K} 4^K = \frac{1}{3}(4^{K+1} - 1)$ which is $O(4^K)$

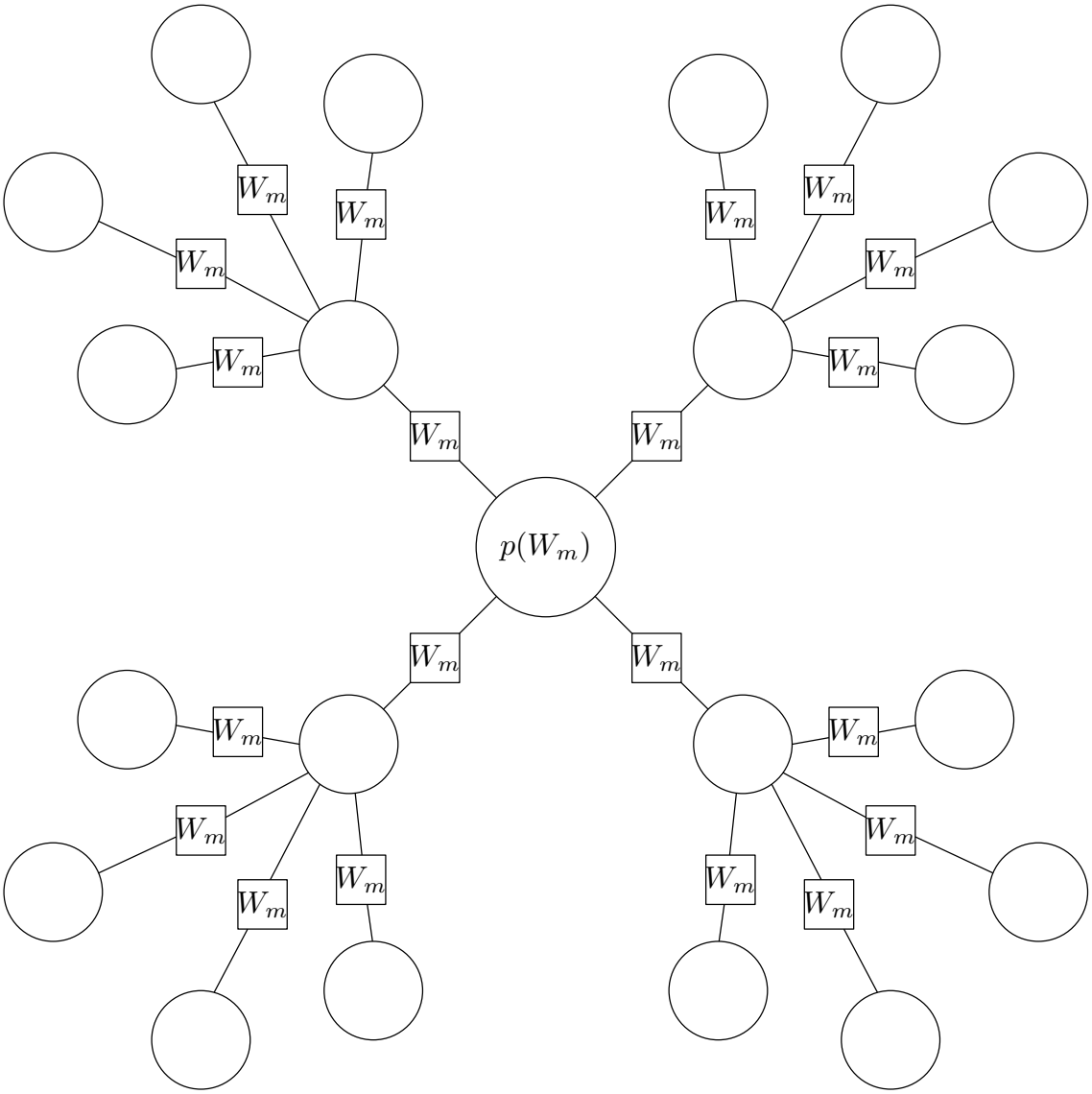


Figure 7.14: Distributed representation of a colour model as a quad tree. All clusters have W_m as their sole cluster variable.

and so $O(N^2)$. This is the same computational complexity as the approximate solution proposed earlier yet, if exact inference were performed, would yield an exact solution.

It was found that the topology of the connections at the bottom of the quadtree affects the quality of the colour models. Figure 7.15 illustrates a naïve connection strategy (reduced to two dimensions for sake of argument) on the left. Neighbouring pixel are merged, following further neighbour mergers until the apex is reached. Because neighbouring pixels tend to have very similar colours, this has the negative effect of causing local "conspiracies", where the importance of certain colours are exaggerated. This is attributed to the damping employed during message passing, where old messages are weighted by 0.1 in new messages, making it progressively more difficult for nodes to communicate as their separation within the graph increases. Instead, using the topology on the right in Figure 7.15 forces disparate regions of the image to communicate strongly.

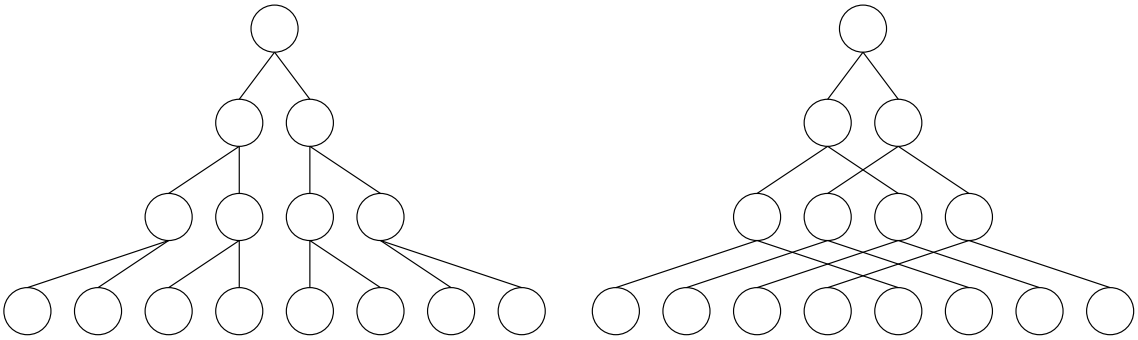


Figure 7.15: Topology of the colour model tree connections. On the left is the naïve method, on the right the method employed in the system. The naïve method allows close communication between neighbouring pixels. As neighbouring pixel colours are strongly correlated, this has the negative effect of nodes "conspiring", leading to pathological behaviour in estimating the colour model underlying markers. Forcing disparate nodes to communicate using the topology on the right avoids this behaviour.

7.10 Message passing

Message passing occurs in the graphical model using the usual loopy belief propagation message calculations [43]. The exact *schedule* by which these occur will now be discussed.

There are four stages of message passing in the graphical model. The first propagates a message from the bottom node in Figure 7.11 to the central node. This enforces prior information related to the value of the pixel. This message is never updated again.

The second kind of message passing starts at the central node in Figure 7.11 and ends at the apex node in Figure 7.14 for each of the colour models. This is called the colour model estimation pass, as it propagates pixel-level information about the colour models upwards to the colour models, updating them in the process. During this pass, old messages retain 0.1 weight in new messages (see "damping" in Koller [43]). This helps to avoid oscillations and convergence to incorrect colour models. Note that the factor of 0.1 is compounded with the traversal of each edge, and so this small damping constant has a large effect as the separation between the individual pixels and the colour model apex increases.

The third kind of message passing is the same as the colour model estimation pass, except it is oriented in the opposite direction. It starts at the colour model apexes, and ends at the individual central pixel clusters in Figure 7.11. This is called the colour model enforcement pass, as it combines prior knowledge of the colour model with propagated knowledge about pixel colour, using these to update individual pixel beliefs. Again, old messages retain a 0.1 weight in new messages.

The fourth kind of message passing considers the pixel neighbour constraints in Figure 7.12. This spatial coherence message passing phase updates edges in series. First, a left to right pass is performed on the horizontal connections. Then an upward pass is performed on the vertical connections. This is followed by a right to left pass in the horizontal connections, ending with a downward pass on the vertical connections. Updating the messages in series allows beliefs based on spatial coherence to spread further than one pixel during this process. This is again

moderated by the weighting (0.1) placed on old messages within new messages.

The schedule of the entire inference process consists of an colour model estimation (upward) pass, followed by a colour model enforcement (downward) pass. Finally, the spatial coherence message passing phase is performed.

During experimentation, it was found that four iterations of this process result in convergence. At this point, a final colour model estimation pass occurs, and the final colour model posteriors are calculated at the apex clusters as in the centre of Figure 7.14.

7.11 Results

The inference process and its results will now be demonstrated using a real example. The spatial priors in Figure 7.6 are applied to the calibration model, with the pixel hue, saturation and value set as constants from the original calibration image (also shown in Figure 7.6).

The inference process is illustrated in Figure 7.16, where the evolution of the marker beliefs over time is shown. The leftmost frames are simply the spatial priors. After each iteration of the message passing schedules, the revised beliefs regarding the underlying markers at each pixel are plotted. Close examination reveals that, after four iterations, the marker beliefs correspond strongly to the actual regions in the calibration image where each marker is located.

The final colour models obtained after these four cycles are shown in Figure 7.17b. Note the compactness of the probability mass in each case. The performance of image segmentation using the prior and the posterior colour models is shown in Figures 7.17d and 7.17e. Here, the image in Figure 7.17c (previously unseen by the colour calibration system) is segmented using the prior and the posterior models. Some imperfections in the segmentation remain, such as a missing region of the index finger. However, it is clear from a comparison of the two segmentations that a considerable improvement has been achieved by the calibration process.

7.12 Conclusion

In this chapter, a means of automatic colour calibration based on inference performed with a graphical model was presented. The model variables, the nature of the potentials present in the network, the topology of the cluster graph, and the schedule by which message passing occurs were discussed. These specifications are sufficient to replicate the results obtained by using standard loopy belief propagation methods.

The calibration system delivered colour models that were of sufficient quality to be used in the remainder of this work. One possible avenue for improvement, however, would be the introduction of a further calibration stage in which the posterior colour models are used to infer improved posteriors from video sequences. Colour model inference for all possible poses would then no longer be based on a single training image, and this may avoid segmentation problems such as those experienced for the index finger in Figure 7.17e.

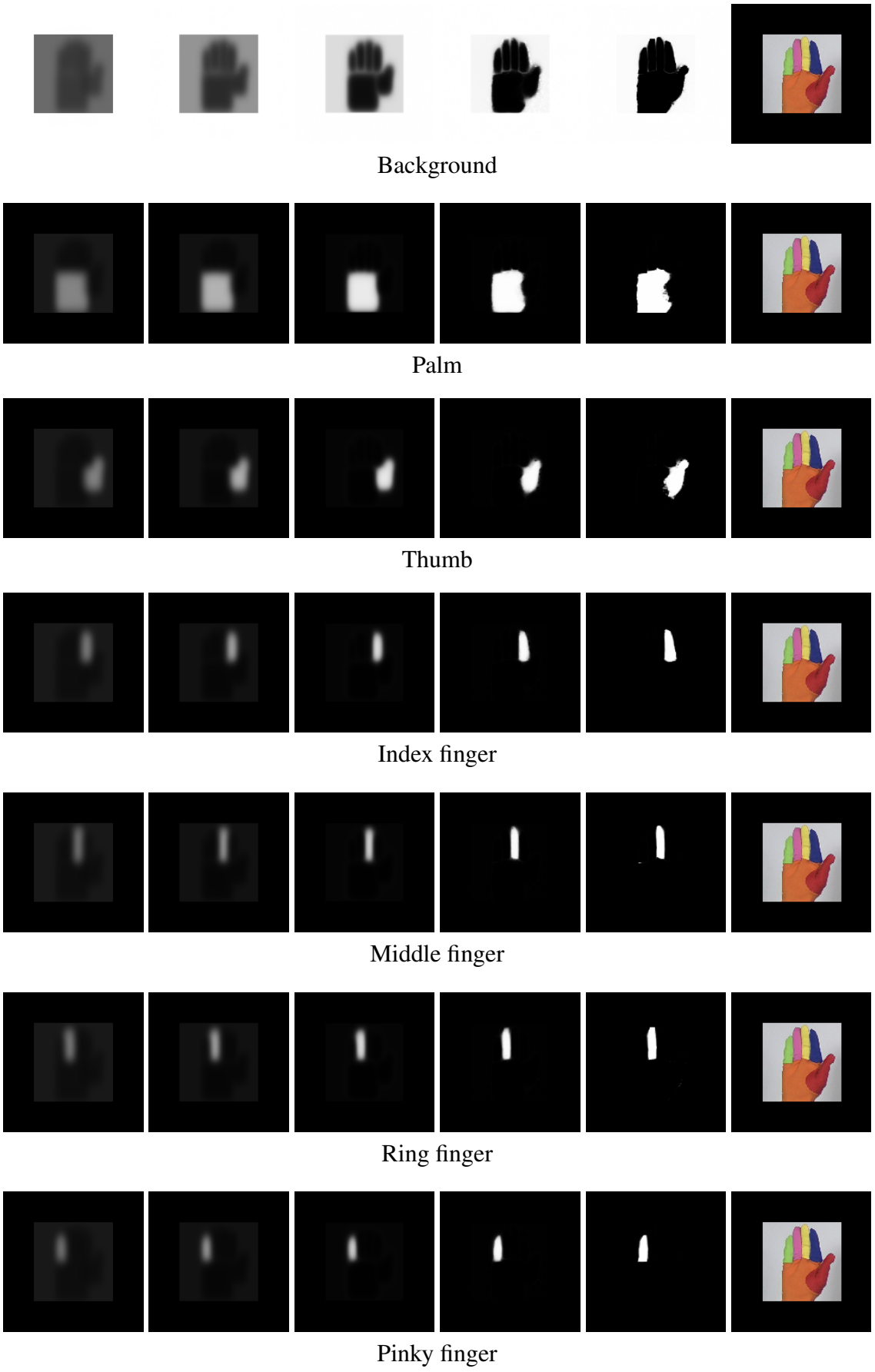
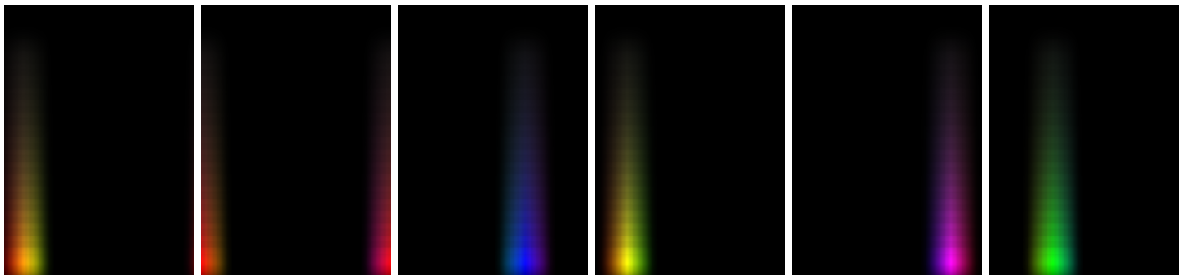
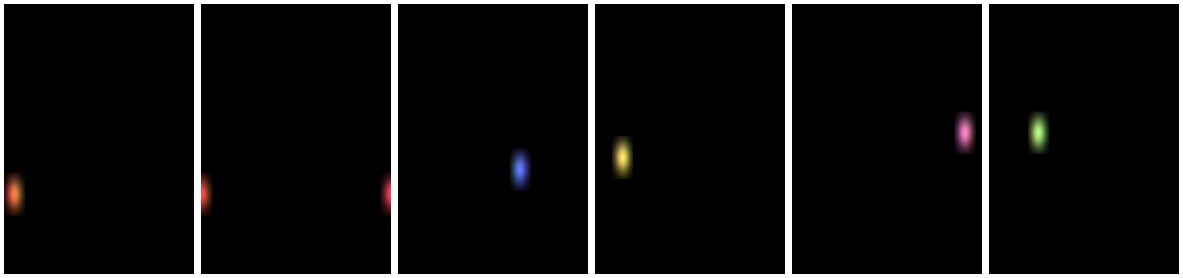


Figure 7.16: Evolution of marker beliefs during the inference process, starting with the spatial prior on the left, until convergence is achieved. The detected glove image is provided on the right for comparison.

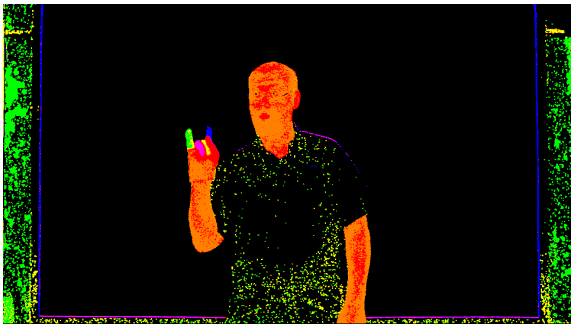


(a)

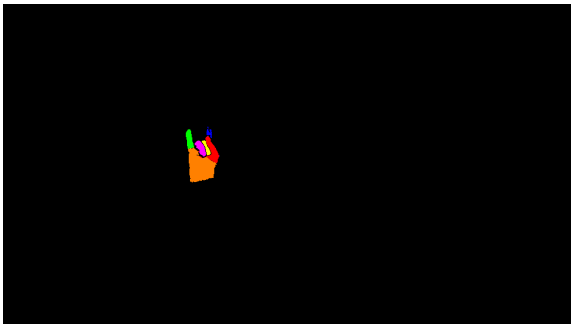


(b)

(c)



(d)



(e)

Figure 7.17: A comparison of prior and posterior colour models: (a) The prior colour models at the start of the inference process. (b) The posterior colour models obtained at the end of the inference process. (c) A test image unseen by the graphical model. (d) The test image segmented using the prior colour models. Note the extensive misclassification of regions. (e) The test image segmented with the posterior colour models. Note that the segmentation is excellent in comparison with the previous case, although the index finger has missing regions.

Chapter 8

Hand Tracking in Video Sequences

The previous chapters discussed the fundamental operation of the pose estimation system. Only single frame pose extraction was considered. In this chapter, subsystems necessary for the experimental work on the tutor subsystem are introduced. A hand region tracker which accommodates two-handed signs is discussed in Section 8.1. Section 8.2 outlines further additions to single-frame pose estimation, which now uses a larger database of candidate poses, and performs a rough search executed on a GPU. Finally, Section 8.3 considers how pose estimations obtained from single frames are combined into pose estimates for entire video sequences.

8.1 Rough hand tracking

The rough hand tracking system's function is to locate the regions of the image occupied by the hands, and track these regions over time. It must also identify which hand is the left hand, and which is the right hand. This entire process is illustrated in Figure 8.1, which shows the running example that will be used.

The process starts with the input image, as shown in Figure 8.1a. This image is then segmented using the colour models obtained from the automatic colour calibration phase. This is illustrated in Figure 8.1b. At this point, likely marker pixels have been identified. However, the image may contain misclassified regions resulting from noise, or the presence of a distractor coloured like one of the markers.

The next stage is predicated on the simple observation that regions containing the hand tend to contain more than one marker colour. Therefore, a grid is superimposed on the image. Within each grid block b the number of marker pixels for marker m is counted, resulting in n_{bm}. The block is then assigned a score τ_b, which is formed from the product

$$\tau_b = \prod_m (1 + n_{bm}).$$

This non-linear score particularly favours those blocks that have n_{bm} large for more than one m. A threshold η is defined, and all grid blocks with scores $\tau_b \geq \eta$ are assumed to contain hand

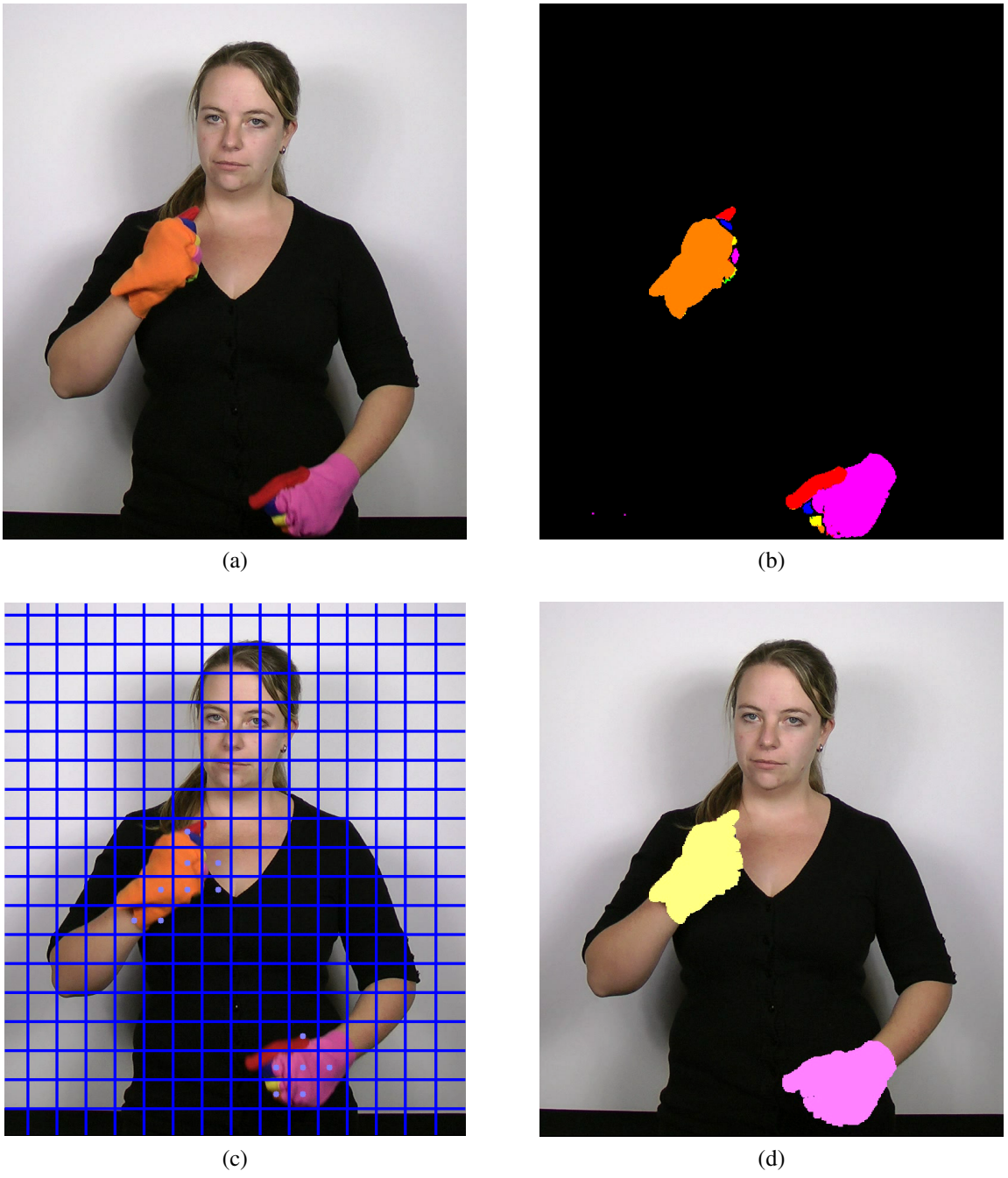


Figure 8.1: Segmenting an image and identifying hand regions: (a) Input image. (b) Image segmented using colour models obtained from calibration stage. (c) A grid is superimposed. Inside each block b, the number of pixels from each marker m is counted forming n_{bm}. A block is considered to contain a hand region if $\tau_b = \prod_m (1 + n_{bm}) \geq \eta$ where η is a preset threshold. This favours blocks where more than one marker is present. Blocks exceeding the threshold are indicated using light blue dots in the figure. (d) The midpoints of the blocks which contain hand regions become seeds for flood fill operations on the segmented image, resulting in a set of candidate hand regions.

regions. This process is shown in Figure 8.1c, where those grid blocks which score greater than the threshold have been marked with light blue dots.

Having identified likely grid blocks which may contain hand regions, the central points of these grid blocks are used as seed points for flood fill operations. The mask on which the flood

fill operations are performed is the result of a bitwise-or between the marker masks obtained from image segmentation. The region connected by a flood fill operation is kept as a candidate hand region, and is removed from the mask so that subsequent flood fill operations will not detect the region again. If a flood fill operation is invoked on a non-marker pixel, no operation is performed.

The result of this operation is illustrated in Figure 8.1d, where likely hand region candidates have been found. Note that, at this stage, it is not known which region represents the left or right hand, or even that a region is a hand at all. After this process, each frame of video has yielded N_i regions R_{in}, where i is the frame number and $n \in 1 \ldots N_i$.

Information from other video frames can be used to find and track hand regions. In the following, it is assumed that the regions extracted from the previous process are either complete hand regions, or not hand regions at all. Then each hand in frame i is assigned to one of the regions R_{in}, or no region in the case where the hand is off camera. Note that the hands may be assigned to the same region in the case of mutual occlusion.

More formally, the assignment of the hands in frame i to regions (or no region) is represented using an ordered pair $S_i = (n,m)$, where n is the region number associated with the right hand, and m is the region number associated with the left hand. A value of zero for either n or m indicates that particular hand is off camera. If R_{i0} is defined as a dummy region that is off camera, $S_i = (n,m)$ indicates that the right hand is part of region R_{in}, and the left hand is part of region R_{im}.

The region assignment S_i can be seen as a state, forming a sequence of states when other frames are included. A score $\sigma(S_i)$ may be defined, which is large for "good" assignments, and low otherwise. A state transition score $\delta(S_i, S_{i+1})$ may be defined, which is large for "good" transitions, and low otherwise. The score of a state sequence with K frames is then

$$\sigma(S_K) + \sum_{i=1}^{K-1} \left[\sigma(S_i) + \delta(S_i, S_{i+1})\right].$$

This score can be optimised using the Viterbi algorithm [51] given the two score functions. The definition of σ and δ in the system embody a number of heuristic rules regarding hand motions. The transition score $\delta(S_i, S_{i+1})$:

- goes to $-\infty$ if one of the hands has moved by more than 10% of the side length of the frame;
- is lowered by 0.01 for each hand that is missing in both the current and next time step; and
- is lowered by 1 if a hand either appears or disappears in the bottom 10% of the frame, but goes to $-\infty$ if one of the hands appears or disappears in the top 90% of the frame.

The region assignment score σ has a more complex set of heuristics associated with it, but the following general remarks can be made about $\sigma(S_i)$:

- if S_i assigns no hand regions, σ gets progressively more negative with an increasing number of large candidate regions available in the current frame;
- if S_i assigns only one hand region, σ gets progressively more negative as the number of candidate regions goes above or below 1;
- if S_i assigns only one hand region, it is slightly biased towards assigning the right hand rather than the left hand, meaning right-handed signing is assumed for one-handed signs;
- if S_i assigns both hands to the same region, a fixed cost is deducted from σ; and
- if S_i assigns two hand regions, then the relative horizontal position of the two hands penalises σ if the right region assignment is to the left of the left region assignment (from the signer's perspective).

After Viterbi alignment, a region assignment for each of the hands is available at each frame of the video. Currently, if hands occlude each other in a frame, the hands are marked missing during that frame. While the set of signs used to demonstrate the tutor does contain two-handed signs, the hands do not closely interact in them, making the frame-dropping acceptable.

8.2 Single-frame pose estimation

Single frame pose estimation occurs as in Chapter 5, but with a few modifications. Firstly, the number of viewpoints is increased. A sphere covering with 130 points due to Sloane [63] is used as the set of viewpoints. All viewpoints with an elevation below $-\frac{\pi}{3}$ are discarded (these are all hand orientations where the palm is pointing backward toward the user), leaving 120 remaining viewpoints.

Furthermore, the image normalisation procedure in Section 5.2.2 is modified. The scaling operation still occurs as is, but rotation is handled differently. The system tests 32 evenly discretised rotations for each pose candidate, selecting that rotation where the areas-and-means metric d^* is the smallest. In effect, this means that the number of possible hand orientations has increase by a factor of about $32 \times \frac{120}{15} = 256$ (recalling that the number of viewpoints was previously 15).

The rough search using the areas-and-means metric d^* is implemented using OpenCL [40] and runs on a GPU. Note that the 32 rotated versions of the areas-and-means features can be derived from a single instance by rotating the mean components by a given amount (the areas remain the same). The rough search can thus be performed quickly. The poses are then sorted according to the calculated distances, and the K nearest neighbours returned for refinement.

At the time of writing the Earth Mover's Distance (EMD) refinement step has not yet been integrated with the updated system, and so the tutor results presented in the dissertation were generated using the areas-and-means metric only.

As the left hand glove is a mirror image of the right hand glove, with palm and ring finger colours exchanged, the existing single-frame pose estimation can be used for the left hand as well. The features are mirrored horizontally, and the palm and ring features exchanged. When the final pose estimate is obtained, its viewpoint must be mirrored horizontally.

8.3 Multi-frame pose estimation

The hands are, at this stage, dealt with separately. After single-frame pose estimation, either it has been decided that a hand is missing in a given frame, or there are K nearest neighbour candidates for its pose.

Let the set of pose candidates at frame i be P_i, with pose candidates $p_k^i \in P_i$ where $k \in 1 \ldots K$. If the hand is marked as missing, there is a single dummy pose candidate p_1^i. Each of the poses can be seen as a possible state of the hand. In order to determine an optimum pose sequence by Viterbi alignment, a score $\psi(p_k^i) = -\alpha_{\text{feat}} d^*(f_i^*, f_{p_k^i}^*)$ is assigned to each pose, where f_i^* and $f_{p_k^i}^*$ are the areas-and-means features associated with the input and the synthetically rendered pose p_k^i respectively. α_{feat} is an experimentally determined weight. This score is an indication of the goodness of fit between the input features and the pose candidate's synthetic features.

A transition score $\tau(p_k^i, p_j^{i+1})$ is also assigned to pose pairs, which is an indication of the likelihood due to physiological considerations that a given pose p_k^i is followed by p_j^{i+1}. This is composed of two parts

$$\tau(p_k^i, p_j^{i+1}) = \begin{cases} 0 & \text{, if either hand is missing} \\ \alpha_{\text{shape}} \varepsilon_f(p_k^i, p_j^{i+1}) + \alpha_{\text{orient}} \varepsilon_v(p_k^i, p_j^{i+1}) & \text{, otherwise} \end{cases}$$

where $\varepsilon_f(p_k^i, p_j^{i+1})$ and $\varepsilon_v(p_k^i, p_j^{i+1})$ are the fingertip error and the viewpoint error between the poses p_k^i and p_j^{i+1} described in Chapter 6. α_{shape} and α_{orient} are experimentally determined weights.

Viterbi alignment may now be used to optimise the sequence of pose assignments $\mathbf{p} = p^1, \ldots, p^M$ where $p^i \in P_i$

$$\mathbf{p}_{\max} = \underset{\mathbf{p}}{\operatorname{argmax}} \left\{ \psi(p^M) + \sum_{i=1}^{M-1} \left[\psi(p^i) + \tau(p^i, p^{i+1}) \right] \right\}.$$

After Viterbi alignment, the final sequence of poses has been determined for the hand, which is then passed to the tutor subsystem.

8.4 Summary

In this chapter, procedures for hand pose estimation in sequences of video frames were discussed. A method by which hand regions can be tracked and classified as right- or left-hand

regions was also described. Viterbi alignment is used to assign labels to candidate regions, and is able to detect occlusion of the hands. Hand pose estimation is performed on the classified regions. The nearest neighbours are fed to an additional Viterbi alignment which selects pose sequences which are physiologically plausible. In the next chapter, conclusions regarding the pose estimation system will be presented. With the hand pose estimation system in place, this is followed by the next part of the dissertation, which details the tutor subsystem.

Chapter 9

Summary and Conclusion for Hand Pose Estimation

In this chapter, the work presented in this part of the dissertation is summarised. The motivation for and the significance of each stage of the research process is discussed. This complements a more compact conclusion presented in Chapter 16, which focusses on the goals, contributions and significance of the entire dissertation.

9.1 Pose estimation

The purpose of this part of the dissertation was to investigate hand pose estimation systems particularly suited for use as a frontend to a sign language tutor. This particular application imposes stringent requirements on the pose estimation subsystem. Hands move rapidly during signing, posing a challenge when tracking the hands over video sequences. In addition, the tutor system requires detailed information regarding the state of the user's hands in order to provide corrective advice.

Database driven pose estimation systems were identified as a family of systems with the potential for providing such information. Such systems provide pose estimates based on single video frames, and therefore recovery from any tracking errors during rapid hand motion occurs gracefully, without the risk of permanent loss of tracking as in model-based approaches. Database driven systems also offer the possibility of high-fidelity hand pose estimates if the database element coverage of the hand state space is sufficiently complete, and so can provide the detailed estimates of pose parameters needed by the tutor subsystem.

One of the most critical parts of a database driven system is the similarity metric employed, as it determines both the theoretical accuracy of the system, as well as the search strategies that can be employed to resolves queries rapidly. During the investigation, the properties of existing systems in relation to the similarity metrics they employ were considered in detail. Certain systems employ simple features like bounding ellipses which, though enabling the system to discard the majority of poor candidates, limits accuracy by not being able to respond to the

detailed shape of the hand. Other systems employ metrics related to the chamfer distance which, though operating on detailed contour information, do not keep account of regions that have already been matched, and so may not be able to preserve information related to contour shape when dealing with misalignments between templates and realistic input images.

The Earth Mover's Distance (EMD) was found to be a promising candidate for addressing these concerns, as it is capable of detailed matching of contours. While body pose estimation based on body silhouette features has previously been attempted with the EMD, it had not yet been investigated in the context of hand pose estimation where contour information is separated according to marker colour. To incorporate contour information from different markers, the compound EMD was proposed in Equation 4.2.2. As with the chamfer distance, the EMD is computationally complex, and so a means of speeding runtime query resolution formed part of the investigation.

Experimental work was performed to address two key questions: would the compound EMD outperform a related system [8] using the chamfer distance and, if so, could it be approximated accurately by using a simpler metric to perform an initial search, followed by refinement using the compound EMD?

To facilitate this work, 500 real images of a gloved hand were recorded with associated ground truth data. This was used to study the pose reconstruction error of different pose estimation approaches quantitatively. By considering the pose error between the ground truth data and the returned K nearest neighbour candidates of the different approaches, it was established that the compound EMD outperformed the chamfer distance in terms of pose estimation accuracy.

However, this result would not be as valuable without a means of efficiently approximating the compound EMD during runtime queries. To this end, an approximation where a simple metric based on the areas and means of the glove marker regions was employed to perform an initial search, followed by refinement using the exact compound EMD. The simple metric can be evaluated rapidly for a large number of candidates on a GPU, further accelerating the approximate search process.

For this approximation to be useful, it was necessary to establish the number of initial nearest neighbours that are needed to provide a good approximation of direct use of the compound EMD, and whether this number is small enough for the resulting search to be feasible. Experimental work determined that 128 nearest neighbours from the initial search were enough to approximate direct usage of the compound EMD closely. Given that the pose database under consideration had 612360 elements, this is a significant reduction in the number of EMD calculations necessary to find nearest neighbours.

The experimental work on pose estimation accuracy did have the limitation of restricting the viewpoint to a subset of those available in the final database. However, the comparison included usage of the exact EMD and the exact chamfer distance, and so the smaller size of the database was necessary to obtain query results in a reasonable time frame. This limitation does have possible implications when considering the number of nearest neighbours needed to accurately

approximate the compound EMD using the two-tier search strategy, which may need to be increased further to maintain the same approximation accuracy. Note that the appearance of the hand changes significantly as its orientation changes, and so the inclusion of all candidates within a limited viewpoint range does include the most likely confusers for a given input image, which may act to mitigate this concern.

The two-tier approach with 128 initial neighbours required, on average, 13s to process a frame of video, and so is not yet usable in real-time. Possibly, a further refinement step using a metric more detailed than the areas-and-means metric, but less computationally complex than the compound EMD would allow the number of compound EMD calculations to be decreased further.

However, the system has improved on a running time on the order of 2200s when direct use is made of the compound EMD, meaning that the system already improves the query resolution speed by more than two orders of magnitude, with high retention of the quality of direct queries. This represents a significant improvement from a system which would be impractical due to excessive query resolution time, to one which can resolve queries within time frames approaching those needed for real-time use. The tutor subsystem chooses a representative frame from each interval of a sign for use in the advice generation process, usually on the order of 5, and so it would be possible to apply EMD refinement only in those representative frames instead of the entire sequence of approximately 500 frames[1]. In this way, the increased accuracy of EMD refinement would be available to the subsystem that requires it most.

9.2 Automatic colour calibration

The pose estimation system presupposes a well segmented image from which to extract contour features. During the initial research process, ad hoc methods such as used by other coloured glove-based systems [2, 20, 80, 91] were employed to define colour models in ways which require a high degree of manual intervention. However, these are not satisfactory solutions, not only because of their impracticality within an end user system, but also because subtle changes between older and newer test sets would require the same manual process to be performed. While constrained region growing techniques such as in [90] exist, the pose estimation system uses detailed contour information, and so simple region growth constraints such as elliptical shape are inappropriate to the task.

A certain amount of prior knowledge is available to the system regarding the colours of each glove marker, but this knowledge is vague. Simply stating that the thumb is “red” hides the uncertainty inherent in such a statement, related not only to the variation within the gamut of colour that may subjectively be referred to as red, but also the context sensitivity of human vision and the effect of the ambient illumination’s spectral properties. Prior knowledge of the marker colours therefore, though available, is inherently uncertain.

[1] Videos from the tutor test set were recorded at 50 frames per second, and had duration on the order of 5s.

Having determined that ad hoc colour calibration procedures involving manual intervention were unsuited to the task, an alternative approach was proposed and investigated which uses a graphical model to reason probabilistically about the prior knowledge available to determine more accurate posteriors. The system operates on a single calibration image where the user is asked to assume a simple hand pose. Few assumptions are made on the exact position and scale of the hand, making obtaining the necessary data simple relative to, for example, manual segmentation of marker pixels.

The essential idea underlying the functioning of the system is to first obtain approximate knowledge about the location of each marker's pixels through the use of rough prior knowledge of the colour models and prior knowledge about the hand pose the user is asked to assume in the calibration image. This first stage uses a fast detector similar to that used in the Viola-Jones face detector [72]. Following this first stage, prior knowledge about the colour models and the marker pixel locations can be combined within a graphical model, and converted to more certain posterior knowledge about the colour models by means of probabilistic inference.

Having established roughly where the marker pixels are located within the calibration image, and using the prior knowledge of the colour models, it was demonstrated that the system could reason about this prior knowledge, and infer with greater certainty the sets of marker pixels and the colour models with each iteration of the message passing schedule.

It was noted that, after colour calibration, the segmentation does occasionally have missing marker regions. This could possibly be attributed to the use of a single calibration image in determine the colour model. One course of action would be to infer better posteriors based on a further calibration video sequence, which would introduce further diversity into the dataset.

However, as was demonstrated in Figures 7.17d and 7.17e, a substantial improvement in segmentation performance based on the posterior colour models relative to the prior colour models is achieved using the system. This relatively clean segmentation allows the pose estimation system to operate on the detailed contour information that the EMD is capable of considering.

9.3 Prelude

Further discussion of the pose estimation system in relation to the entire system is given in Chapter 16, which includes a summary of key contributions and an exploration of future work that may flow from the current research. In the next part of the dissertation, the tutor subsystem, its theoretical underpinnings and its evaluation on real data are presented.

Part II

Sign Language Tutor

Chapter 10

Introduction to Sign Language Processing

In the following chapters, a context-sensitive advice generation system for a South African Sign Language tutor is presented. The advice is based on linguistic descriptions of the constraints that the user must obey. The domain-specific language with which constraints are specified allows arbitrarily complex compound constraints to be created from atomic constraints as well as other compound constraints.

Despite the potential complexity of the compound constraints, the system automatically reduces these to a form with a simple logical structure. This final form of advice is based on a simple conjunction of atomic constraints, which means that the user need not mentally "parse" a complex logical structure in order to understand the feedback from the system.

The system can be extended fairly easily. New atomic constraints can be defined (indeed, the system can be used to give advice in problem domains other than sign language teaching). The exact form of advice (examples might include textual or graphical feedback) can be defined freely for each atomic constraint. The advice mechanisms are provided with both relevant observed features and a suggested set of features which would fix the constraint violation, allowing the advice to be dynamically tailored to the context of the user's current attempt and the most parsimonious way of correcting their mistake.

Before describing this system, a discussion of related work will be given. In Section 10.1, an overview of natural language processing of South African Sign Language in particular is provided. This is followed by Section 10.2, which describes the fields of gesture and sign language recognition, and how they relate to sign language tutors. In Section 10.3, existing sign language tutors are discussed and classified. Special attention is paid to the types of feedback these systems are capable of producing. This is followed by a general appraisal of the state of the art. Finally, the content of subsequent chapters is outlined.

10.1 South African Sign Language

South African Sign Language (SASL) is an under-resourced language, and its fundamental linguistic aspects are still a subject of ongoing research. There are, however, a few instances

where SASL has been the subject of natural language processing systems.

Rajah [52] presented a recognition system for SASL that operates on features due to Rigoll et al. [54], which are derived from the image moments of frame-by-frame difference images. Hidden Markov Models are employed to recognise words from a lexicon of 17 SASL signs.

Naidoo and Connan [49] describe an HMM-based recognition system for SASL which tracks unadorned hands. The dimensions and position of the head are used as a reference point for locating other body parts using fixed anatomical ratios. Using the position of the hands relative to these body parts as features, they train HMMs and perform classification tests using a 10 word lexicon.

Achmed and Connan [1] compare two approaches for upper body pose estimation and perform recognition experiments on a lexicon of 20 SASL words. The first approach is a database driven system, similar to that which Athitsos and Sclaroff [8] proposed in the context of hand pose estimation. Synthetic posture candidates are stored during a precomputation phase. At runtime, the candidates are searched, using the chamfer distance on silhouette to find the closest database candidate to the input. This system is compared with a second approach based on Support Vector Machines (SVMs) trained with a set of exemplars. The authors report an increase in correct pose estimation from 61% to 81% between the first and the second approach.

Efforts toward machine translation of SASL include Van Zijl et al. [69–71], who report on a rule-based machine translation engine for the translation of English to SASL. Bungeroth et al. [19] present results obtained from a statistical machine translation engine on a corpus that includes SASL sentences related to air travel information services.

10.2 Gesture and sign language recognition

When specifically considering sign language tutors that monitor the user's signing, a natural association emerges with the fields of gesture and sign language recognition. Indeed, many of the techniques used to create sign language tutors stem from previous application in these fields.

In this section, an overview will be provided of gesture and sign language recognition. A representative sample of such systems will be discussed. In particular, the variability in key characteristics of gesture and sign language recognition[1] systems will be discussed. These characteristics include: the input method, granularity of smallest recognisable unit (phoneme-level or word-level), the scope of the largest recognisable unit (fingerspelling, word-level or sentence-level) and the recognition architecture (for example, neural networks or HMMs). Each of these aspects will be discussed in turn in the following subsections.

[1] Except where a distinction is necessary, "gesture recognition" will be used inclusively to refer to both gesture and sign language recognition.

10.2.1 Input methods

The input methods used for gesture recognition have changed considerably over time, shifting gradually from the use of data gloves [92], to vision-based methods that require gloved hands, to relatively unrestricted vision-based systems.

Early systems commonly used data gloves to collect data about the shape of the hand directly. The position and the orientation of the hand, if used by the system, would be measured by attaching sensors to the gloves. Examples of such systems include [31, 41, 48].

This type of data collection has the advantage of being able to directly measure quantities of interest such as hand position, velocity, orientation and joint angles. It was common for recognition systems to forgo further feature extraction in favour of simply using these parameters directly during recognition. Even one of the earliest examples of such systems, described by Murakami and Taguchi [48], had direct access to hand position, orientation and finger joint angle information.

Data gloves are, however, expensive and constrict the user's movements. There is also no way for such systems to collect data regarding non-manual gestures without the addition of, for example, a camera.

More recent work tends to focus on vision-based methods, though the exact nature of input shows considerable variation within this category. Because the information obtained about the user's movements is now relatively indirect, the process of pose estimation is no longer trivial. Earlier examples of such systems [64] tended to rely on the use of coloured gloves to ease the tracking of the user's movements. This requirement has relaxed, with many current systems tracking unadorned hands [36, 84, 88].

However, equating unadorned hand tracking with the state of the art is misleading. Relatively recent work [36, 84] may track unadorned hands, but the positional and ellipse-based features on which they ultimately operate are similar to the ellipse-based features in Starner and Pentland's [64] early glove-based system. By contrast, state of the art sign language tutors of Sagawa and Takeuchi [58], Zieren [91] and Zafrulla et al. [87] use gloved hands (Sagawa and Takeuchi use data gloves; the latter two systems use coloured gloves).

These trends reflect a fundamental difference between gesture recognition and sign language tutoring. Sign language tutors need as much state information about the user's hands as possible in order to judge their efforts, whereas the recognition problem does not necessarily require a detailed reconstruction of hand pose. This is not to say that detailed pose information would not be useful for gesture recognition. Instead, indirect information about the hand pose present in the *appearance* can be processed into a variety of features which prove to be more robust for the recognition task than attempting detailed pose estimation. However, because these features do not capture hand pose information directly, they are not immediately useful for sign language tutors.

10.2.2 Granularity and scope of recognition

In the general case, gesture recognition is aimed at recognising either isolated gestures, or sequences of gestures. For the task of sign language recognition, the ultimate goal is to recognize words as they occur in sentences. As such, the smallest units to be recognised are sign language words.

However, it is possible to recognise even smaller units. As with orally spoken words, sign language words have an internal structure. The building blocks of a sign are discrete and well-defined, and are meaningless in isolation. These building blocks are the equivalent of the phonemes of orally spoken languages, and are currently also referred to as such (previously, they were called cheremes). Sign language phonemes include aspects [47] such as:

- hand shape;
- hand position;
- hand movement;
- hand orientation; and
- non-manual gestures such as facial expressions.

There are sign language recognition systems that attempt to capitalise on the phonology of these language [15, 44, 75–77], though this is by no means essential. However, the substructure of sign language words becomes important when a system must act as a *tutor*. Without an awareness of the phonology of a word, it is impossible to comment on the correctness of a signing attempt beyond an estimate of the degree of correctness.

Another aspect to consider is the largest unit that may be recognised by a system. Recognition systems may be classified into three groups.

The first group are those systems that recognise only static poses. Systems that recognise fingerspelling are a particularly limited form of this type of system. Fingerspelling is a form of signing which is used to spell words or names taken from an oral language, using hand poses. Each pose corresponds to a letter from the alphabet of the oral language. These hand poses are usually static (although occasionally they may have dynamic aspects), and so are a relatively easy target for recognition systems. In isolation, their applicability to sign language recognition is limited, as they do not address the words native to the sign language itself. Examples of such systems include [3, 42].

As Kim et al. [42] note, however, word- and sentence-level sign language recognition systems tend to focus on the "larger" aspects of the signer's movements (this is related to the earlier discussion in Section 10.2.1 on the features employed by contemporary sign language recognition systems). Fingerspelling may be difficult to recognise for such systems. While not applicable in isolation, fingerspelling recognition systems therefore have a complementary role to play with respect to sentence level sign language recognition.

Systems in the second group recognise whole, but isolated, words which may include movement. Examples of such systems (discussed in more detail in Section 10.2.3) include [31, 35, 37, 41, 48, 73, 74, 90]. At this level, useful applications such as simple dialog systems and word-level translation become possible. However, as mentioned earlier in this section, the applicability of such methods to sign language tutors is strongly dependent on whether the substructure of words is analysed explicitly during recognition.

Systems in the third group perform sentence-level recognition. Section 10.2.3 discusses several examples of such systems [36, 37, 41, 64, 65, 75–77, 84]. Automata based methods, such as HMMs, are a natural choice for sentence-level recognition, as they can be used to model the underlying grammar of the language. Isolated word recognition does not necessarily imply easy extension to sentence-level recognition. Continuous sign recognition is partly hampered by a form of coarticulation known as movement epenthesis [33], during which meaningless portions are inserted before and after a sign due to the neighbouring words in the sentence. Systems which do not consider word substructure, but do support sentence level recognition, may still be applied to sign language tutoring (see Zafrulla et al. [87]). Such systems may teach aspects of grammar, and help users practice their skills at producing entire sentences.

10.2.3 Recognition architecture

The actual subsystem which is responsible for the classification and recognition of signs is of particular importance. The sign language recognition field has extensively borrowed techniques from the field of oral speech recognition. Among the different recognition paradigms that have been employed, artificial neural networks (ANNs) and hidden Markov models (HMMs) are prominent examples. Because the exact details of how these methods are employed vary considerably, a representative selection of existing sign language recognition systems will now be presented, classified by recognition paradigm. Special attention will be paid to HMMs, as sign segmentation in our tutor is performed with an HMM generated from the domain specific language (see Chapter 13).

10.2.3.1 Artificial neural networks

ANNs were employed by some of the earliest sign language recognition systems. The recognition of particular static hand poses is a typical task performed by such systems. Certain systems such as proposed by Al-Jarrah and Halawani [3] are limited to isolated hand pose recognition such as fingerspelling recognition.

Word-level recognition, if it is attempted at all, either requires either unified dynamic network architectures or ad hoc methods to process movement information. Unified architectures include recurrent neural networks, such as those presented by Murakami and Taguchi [48], or the 3D Hopfield network described by Huang et al. [37].

If the recognition network architecture does not support movement explicitly, ad hoc methods must be employed. This might involve an additional neural network to assist in movement segmentation, such as in Fels and Hinton [31] where a separate "strobe" network signals the moment at which the movement of the hand should be sampled. Other ad hoc approaches do not use ANNs, such as the system discussed by Kim et al. [41] who use a lookup table based approach driven by quantised movement information.

An interesting means of handling dynamic information is to embed neural networks inside HMMs. The HMMs then segment movement, while the embedded neural networks process features into more informative representations. Kim et al. [42] provide an example of such a system, investigating the recognition of fingerspelling sequences from videos using a tandem approach [28]. Multilayer perceptrons (MLPs) are trained using SIFT [46] features to either classify letters or, in a novel approach, phonological features of American Sign Language fingerspelling. The MLP outputs act as features for HMMs, which perform the final recognition.

The previous discussion suggests that it is the lack of compelling solutions to handling dynamic motion within signs that hampers efforts to employ ANNs, in isolation, to sign language recognition. HMMs, however, offer a natural means of dealing with temporal structure, and so have assumed a dominant role in the field, as will be presented in the following section.

10.2.3.2 Hidden Markov models

Hidden Markov models have a long history of application to gesture recognition and sign language recognition in particular. Sign language recognition is maturing in the direction of sentence-level recognition, a departure from general gesture recognition. However, early gesture recognition systems using HMMs were particularly important in the development of sign language recognition using HMMs, as the ideas underlying both areas developed together initially.

Conventional HMMs were established as a means of recognising discrete human actions by work such as presented by Yamato et al. [83] who created a system for recognising 6 different tennis strokes, operating on features derived from binary masks representing the location of the human within the image sequence. Subsequently, variants of conventional HMMs specifically tailored to the multiple streams of information inherent in human movement were explored, for example, by the work of Brand et al. [18] on coupled HMMs for gesture recognition. This theme continues within the field of sign language recognition, such as in the system presented by Vogler and Metaxas [75] which utilises parallel HMMs.

HMMs have proven to be effective in the recognition and segmentation of both isolated signs and whole sentences. An early example of a sign language recognition system is described by Starner et al. [64, 65]. This vision-based system is capable of recognising full American Sign Language (ASL) sentences constrained by a simple grammar using a lexicon of 40 words. Since then, a great variety of systems utilising HMMs have been reported in the literature.

Vocabulary size and sentence level recognition are two key aspects that have enjoyed attention within the sign language recognition field. For experimental systems, improvements to either of these two aspects tend to exclude the other. In testing sentence level recognition, it is convenient to constrain the vocabulary so that the datasets can adequately cover the space of possible *combinations* of these words. For large vocabulary recognition, aspects such as the scalability of the system and addressing inter-word confusion are attended to in more detail.

An early example of a relatively large-vocabulary recognition system is that described by Grobel and Assan [35]. The system recognises isolated words using HMMs from a vocabulary of 262 words. As the user wears a coloured glove on the dominant hand which marks each finger with a different colour (including a marker on the back of the palm), a relatively rich feature set was available for classification. This enables the system to separate minimal pairs which differ only in either hand orientation or form. Features extracted from input images include the hand locations, the inter-centroid distances for all markers, the areas of the markers, the distance between the two hands and a measure of the dominant hand orientation based on the polar form angle of each marker centroid relative to the hand centroid. Another large-vocabulary system is reported by Zhang et al. [90], who focus on the scalability of their recognition system by employing tied mixture parameters [12]. The lexicon of the system contains 439 isolated Chinese Sign Language words. In addition to a set of features similar to Grobel and Assan [35] include the eigenvalue ratio of each marker (effectively an indication of the eccentricity of an elliptical approximation of each marker).

Sentence level recognition systems include those by Holden et al. [36] and Yang and Sarkar [84]. The systems employ relatively simple features, based on elliptical approximations of the hand regions, though with the advantage that the hands are unadorned. As the system vocabularies are relatively small (21 words and 39 words respectively), the simplicity of the features poses less of a risk in terms of confusion based only on hand shape and orientation. The system of Yang and Sarkar [84] is particularly interesting for performing the tracking and recognition simultaneously by employing a novel HMM decoding algorithm.

Sentence level recognition can be hampered by movement epenthesis, which are movements between signs which do not convey meaning. Certain systems attempt to improve recognition by modelling such movements explicitly. For example, Fang et al. [30] train transition movement models using a temporal clustering technique based on dynamic time warping.

Certain authors attempt to exploit the phonetic substructure of signs in order to improve recognition. These systems are discussed in the following paragraphs.

After earlier work on word-level ASL recognition [73, 74], Vogler and Metaxas [75–77] proposed a segmentation strategy based on the Movement-Hold phonological model of Liddell and Johnson [45]. Here, signs are composed of alternating stages of movements or holds. They therefore train HMMs for each separate phoneme, rather than entire signs at a time. In addition to using the instantaneous features present at a point in time, they also define “global” features derived from the covariance matrix of the hand trajectory over a fixed time interval, allowing

straight line or arc movements to be segmented. They also train HMMs to detect movement epenthesis between signs. In order to deal with the simultaneous aspects of ASL, they advocate the use of Parallel HMMs [75, 77], which are effectively normal HMMs used independently on different simultaneous feature streams.

Another approach which attempts to examine phonetic substructure is demonstrated by Bowden et al. [15], who make use of a two-stage classifier where raw features were first classified into high-level "viseme" representations (on the level of "hands move apart" or "right hand on left shoulder"), which are then used as features for a bank of Markov model classifiers.

Kong and Ranganath [44] use a rule-based trajectory segmentation algorithm to extract candidate phoneme instances. Principal Component Analysis (PCA) is used to differentiate linear movements from curves. A descriptor of the candidate phoneme is built describing the geometric properties of motion, as well as using a Fourier descriptor to encode shape. K-means clustering is used to find phonemes, which are used to train HMMs for continuous sentence recognition.

HMM-based systems like those systems discussed in this preceding paragraphs, demonstrate the utility of exploiting phonological structure in sign language recognition. While their primary task is not that of segmenting signs, HMMs segment signs into phases as a byproduct of the recognition task. It is this aspect of HMMs that is applied to segmenting user signs into different phases in Chapter 13.

Having discussed sign language recognition systems, the focus now shifts to existing sign language tutors, which will be presented in the next section.

10.3 Existing sign language tutors

Several approaches exist for the e-teaching of sign languages. A common one, often encountered in the commercial arena, is to make a dictionary and/or phrase book available to the user, who can see signing demonstrated by means of prerecorded video sequences. The tutor component of such systems tend to ask the user to repeat certain signs or phrases by themselves, offering no feedback other than to display the correct sign when prompted. For South African Sign Language, the SignGenius [60] system is an example of such a tutor application. A further example may be found in Xu et al. [82], who describe a system which tests the receptive skills of children learning American Sign Language (ASL). The system tests user competency in two ways. During the first type of test, the user is presented with a video of a sign, and asked to choose the correct definition from a set of four options. The second type of test considers the inverse problem, that of being given the definition of a sign, and asked to select the video of a sign matching that definition from four possible choices.

Alternatively, such systems may use text-to-speech methods to present signing to the user. Karpouzis et al. [39] describe a system which synthesises animations of Greek Sign Language (GSL) sentences using an on-screen avatar. This is used to teach children aspects of GSL

grammar, though any attempted signing by the user is not monitored.

Of more interest are those systems which monitor user movements by some means to provide feedback about signing correctness. Amongst these systems, the tasks the user is presented with, as well as the types of feedback given, vary considerably.

Perhaps the most basic task a user can be given is to sign some predetermined word or phrase. Systems encountered in the literature tend to revolve around this task, although they exhibit marked differences in what sort of feedback they can provide to the user. It should be kept in mind that, for such systems, the correct sign is known a priori, and so the problem is one of providing commentary based on a specific sign, and *not* classification between different possible signs.

Existing tutors can be grouped according to the way in which the user's attempt is evaluated. These groups are classification-based systems and template-based systems. Classification-based systems are of relatively limited applicability, though they do have a potential role to play in sentence-level tuition. Template-based systems are more interesting, although their ability to provide commentary is also restricted. These two groups will now be discussed in more detail.

10.3.1 Classification-based systems

Classification-based systems make direct use of techniques from sign language recognition to act as tutors. These systems recognise either isolated signs or phrases using classifiers such as a set of Hidden Markov Models (HMMs). They are therefore limited to determining if a given word is present or not, or whether another word has been substituted in its place. More advanced systems may recognise entire phrases, although the word-level granularity remains.

Because classification-based systems are not applicable to cases where the correct sign is known a priori, the teaching paradigms they support are limited. The simplest, though least useful, is to test for the *confusion* of two signs. The systems demonstrated by Aran et al. [4, 5] serve as examples of this teaching paradigm (other examples of such systems exist, with similar limitations).

Aran et al. [5] demonstrated a classification-based tutor system for Turkish Sign Language which makes use of a bank of HMMs to determine whether the user has signed the required word from a lexicon of seven words. The system is vision-based, and uses uniformly coloured gloves to mark the user's hands. While this system is able to detect whether a user has mistakenly confused two signs with each other, it is unable to comment to what extent the required sign has been performed correctly. The system also assumes that one of the words in the lexicon has been signed, and so cannot detect when an utterance falls outside the lexicon.

In [4], Aran et al. report on a more complex system which, in addition, tracks the movement of the head. Classification using HMMs is again employed. In addition to reporting whether the required sign has been performed, the system also features an avatar-based replay of the user's head motions (hand motions are animated from a library), although it does not comment on the correctness of the motions.

Another possible teaching paradigm supported by classification-based systems is developing the *sentence-level proficiency* of the user. In this role, classification systems have more potential. Though they still cannot comment on the correctness of signing given the particular sentence, it is possible to provide an environment where the user has to make a choice regarding the most appropriate utterance to make. This situation is a natural fit for classification approaches, although the system described Zafrulla et al. [87] appears to be unique in this respect. This teaching paradigm is complementary to, and not in competition with, the intra-sign advice generation considered in this work.

Zafrulla et al. [87] outline the development of a game called CopyCat for children to teach them aspects of ASL. The user wears a pair of uniformly coloured gloves, with each glove having a three-axis accelerometer. The user is presented with a simple scene, and asked to describe it. A tutorial video is provided, demonstrating the sign if the user needs help. The system is based around sentence-level recognition of ASL using HMMs with a highly constrained grammar allowing 59 different phrases using a 19 sign lexicon. They employ a rejection threshold (common to all classes) on the normalized log likelihood score to balance the true- and false-positive rates. Similarly to Aran et al., the system is not able to comment on the correctness of the signing components.

10.3.2 Template-based systems

Of more interest are tutors which analyse the correctness of general aspects of the user's signing by comparing the user's attempt with a template provided by an experienced signer or a generated set of animation parameters. The systems described by Zieren [91] and Sagawa and Takeuchi [58] were the only systems of this type encountered in the literature.

Zieren [91] demonstrated such a vision-based tutor system which tracks a user's gloved hands, attempting a detailed reconstruction of their dominant hand pose over time. The user's input is then synchronised with a prerecorded exemplar. Next, the input sign is evaluated by checking the positions and movements of both hands, and the pose and rotation of the dominant hand. The system reports whether there are significant deviations from the exemplar in one of these facets.

Although such systems provide feedback, they lack the ability to express high-level movement constraints, and are limited to making general statements regarding the closeness with which the user matched the template movement. It is also difficult to specify what is *not* important in the template movement in all but the most general terms (perhaps, for example, by choosing a slacker bound on positions or velocities relative to the template).

Sagawa and Takeuchi [58] describe a tutoring system for Japanese Sign Language that uses datagloves to track the movements of the user's hand. They use a knowledge-based system to generate and score sentence hypotheses for the user's input. Feedback generation is limited to sentence-level scoring and synchronisation of the user input with the sentence description

and its animated representation. The user is left to evaluate the differences between their own attempt and the synchronised template.

The system described by Sagawa and Takeuchi [58] represents signs as sequences of simple constraint conjunctions (conjoined by the 'and' logical connective) and does not support more complex constraints. The constraints also cannot be used for anything other than scoring. Without this ability, the facilities to build new constraints are limited, and hard coding new compound constraints is often necessary. This has the negative effect of obscuring the internal structure of these constraints, which makes it harder to provide advice to the user when a constraint is violated.[2] The user may be informed that a constraint has been violated, but is not provided with an indication of what action should be taken relative to the user's attempt. If the internal structure of higher-level constraints were retained, one could, in principle, diagnose which portion of that constraint has been violated and provide targeted advice to the user.

10.4 Discussion

Upon reflection, the obstacle to adequate advice generation in existing systems is twofold. Firstly, the granularity with which the systems evaluate the user's movements is inappropriate for the task. Classification-based systems operate at a level where entire words are the smallest unit. In effect, each word is a single indivisible constraint, and so intra-word commentary is not possible. Template-based systems, while superior to classification-systems in this regard, operate at too fine a granularity. Each frame of video is matched with a particular moment of the template, which acts as a micro-constraint. At this level, where all high-level meaning is lost, it is unsurprising that the constraints on the user are modelled as simple proximity within the feature-space.

Secondly, these systems lack the ability to express the complex logical relationships that may exist between the constraints on the user's movements. Classification-based systems lack this ability because there are no subconstraints which can enter into relationships with each other. Template-based systems lack this substructure because the template features form frame-by-frame indivisible constraints.[3]

These characteristics stand in contrast to what is known about the phonology of sign languages, as discussed in Section 10.2.2. The role of an advice generation system is to comment on the phonological aspects of a user's signing. In fact, beginners are often unaware of the phonology of sign languages, and a tutor should attempt to instil in them an awareness of this fact.

Sign language recognition systems may operate successfully without explicitly taking a language's phonology into account. However, tutors can scarcely avoid explicit reference to

[2] Retaining the ability to treat groups of constraints as a unit is still important. The domain specific language supports this using the `feedback-group` construct.

[3] Note that, while Sagawa and and Takeuchi [58] do have a type of constraint specification, the constraints are associated with animations which act as templates.

language phonology. It is exactly evaluation and advice in terms that a user can understand that is desired, yet currently tutors lack the means of reasoning about a user's attempt on this level. To remedy this situation, we must be able to reason about the phonemes that a word is composed of, and how well the different aspects of a signer's attempt match with what is required of them.

It is natural, therefore, to think of phonemes as discrete constraints. This is the level of granularity at which feedback is needed. To provide truly context-sensitive feedback, therefore, a constraint-based language is needed with which one can reason about a user's attempt. This language must be sufficiently flexible to accommodate the different types of phonemes, and be easily extensible when new constraints are necessary. Furthermore, it must be possible to accommodate the potentially complex logical interactions of the constraints. Simple and-semantics is insufficient for this task, and hence the addition of other logical connectives such as 'or' and 'not' will be necessary. The description language that embodies all these considerations will be defined in Chapter 11.

The ability to describe constraints is not sufficient for tutor systems, since it must be possible to use the language to reason about the user's attempt, identify problems, and then finally to provide targeted advice with which mistakes may be corrected. Due to the introduction of additional logical connectives, the constraints placed on a user's movements may be complex. Disentangling such structures into correct advice that the user may readily comprehend, is non-trivial. Chapter 12 presents a procedure with which this may be achieved when given a single frame of video and a corresponding set of constraints.

Ultimately, the tutor system must generate advice for entire video sequences. Therefore, a way is required to describe the constraints applicable to each phase of a sign, and a means of determining to which phase of a sign each frame of video belongs. Advice is then generated based on single frames selected from the video sequence. This process is detailed in Chapter 13.

This system is implemented and evaluated in Chapter 14 on real video sequences of a selection of South African Sign Language signs. A variety of constraint types, and the advice that is generated from them, are demonstrated.

Finally, Chapter 15 reviews the contributions made to the state of knowledge around the e-teaching of sign languages, and concludes this part of the dissertation.

Chapter 11

Features and Constraints

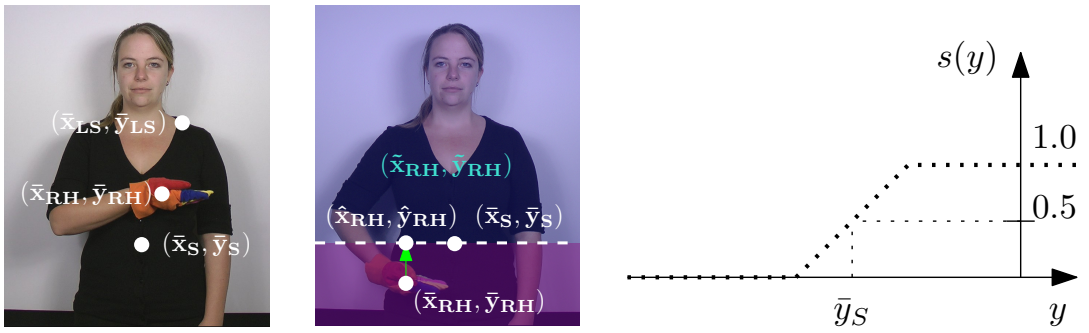
We start our discussion of the tutor system by discussing its simplest components, namely features and atomic constraints. Following this, we will show how compound constraints can be built by combining atomic constraints. The SASL sign for "woman" will be used as a real-world example to illustrate these concepts. Figure 11.1 shows key frames from a correct attempt of this sign.

Figure 11.1: The three intervals for the SASL sign for "woman". The right hand is held palm facing upward, fingers pointing leftward. It briefly touches just below the left chest, moves rightward and then briefly touches just below the right chest.

11.1 Features

Features retain their usual meaning from the world of machine learning, in that they are quantities which capture information about the current state of the world. Features may, for example, be the horizontal and vertical positions of the hands, or the amount of bending in the index finger. In Figure 11.2a, the positional features of three body parts are illustrated.

Each frame of video, after analysis, produces a set of features which become the input features for the tutor. We will denote the set of all the input features for a given frame of video as $\bar{x}$. These features are used for the initial evaluation of the user's movements. If it is found that the user has made a mistake, the input features are used as a starting point for the advice generation. The system then tries to find "nearby" features that will correct the mistake.



(a) (b) (c)

Figure 11.2: (a) Six positional features within a single video frame taken from a correct signing of the SASL word for "woman". The features represent the x- and y-locations of three body regions: the right hand $(\bar{x}_{RH}, \bar{y}_{RH})$, the left shoulder $(\bar{x}_{LS}, \bar{y}_{LS})$ and the stomach $(\bar{x}_S, \bar{y}_S)$.
(b) Positional features where $\bar{y}_{RH}$, in the light of the constraint `(higher-than stomach right-hand)`, would be considered incorrect . The legal values of the constraint variable $\tilde{y}_{RH}$ are those in the region above the dashed line. The suggested feature $\hat{y}_{RH}$ would be the valid value of the constraint variable $\tilde{y}_{RH}$ closest to the input value $\bar{y}_{RH}$ as indicated. It can be seen that this point corresponds to the smallest possible movement of the user's hand to correct the error.
(c) The satisfaction function $s(y)$ of the constraint `(higher-than stomach right-hand)` where $\bar{y}_S$ is the height of the stomach obtained from the input, and y is the height of the right hand. For any value of y greater than $\bar{y}_S$, $s(y) \geq 0.5$ and thus deemed acceptable. This corresponds to the region above the dashed line in (b).

We use the center of the face as the origin, and the width of the head as the unit of length. The face is detected using the standard Viola-Jones detector available in the OpenCV library [17]. Approximate values for the input features $(\bar{x}_{LS}, \bar{y}_{LS})$, $(\bar{x}_{RH}, \bar{y}_{RH})$ and $(\bar{x}_S, \bar{y}_S)$ in Figure 11.2a might be $(-1, -1)$, $(-0.5, -3)$ and $(0, -5)$ respectively. Note that the coordinate system is defined from the user's perspective. This means that the axis pointing rightward, the positive x-axis, is oriented to the left within the images.

The location of various body parts are estimated using fixed proportions relative to the width of the head. This is similar to the method used by Naidoo and Connan [49], and proved sufficient during subsequent testing.

The need to generate advice introduces two other roles for feature variables. The set of input features $\bar{x}$ has a corresponding set of constraint variables $\tilde{x}$ which interact with constraints (constraints will be defined shortly). The constraint variables represent the world as it should be. Initially, each constraint variable is allowed to vary over all its possible values, but this variation is progressively limited by application of the constraints associated with a sign.

Constraint variables, after the application of constraints, represent the possibilities open to the user for correctly doing the sign. All legal values of the constraint variables represent valid alternative ways of performing the sign. The constraint `(higher-than stomach right-hand)` would, for example, limit $\tilde{y}_{RH}$ to be above -5, whereas before the application of the constraint -6 might have been a legal value. This is illustrated in Figure 11.2b where the legal values of $\tilde{y}$, after the application of the constraint, are those above the dashed line.

If the user's attempt was incorrect, the input features $\bar{x}$ will not be a legal value for the constraint variables $\tilde{x}$, and a new value must be found. This new value, the suggested value $\hat{x}$, represents the nearest point at which the user can reach the world as it should be ($\tilde{x}$) from the world as it currently is ($\bar{x}$).

The suggested value is one particular choice of the legal values of the constraint variables $\tilde{x}$, but the selection of this value is tempered by the desire to demand the least effort necessary of the user. The system attempts to find that set of features $\hat{x}$ which will be easiest for the user to target, based on their current attempt $\bar{x}$. This is done using a procedure that simultaneously minimises a measure of the user's effort while ensuring that the suggested features $\hat{x}$ will be a legal value of the constraint variables $\tilde{x}$. A detailed discussion of this procedure is presented in Section 12.4.

Figure 11.2b shows an example of finding a suggested value, where the closest legal value of $\tilde{y}_{RH}$ is $\hat{y}_{RH}$, which corresponds to the smallest movement the user's hand can make in order to correct their mistake.

11.2 Atomic constraints

Features describe the state of the world as it is (the input features $\bar{x}$), as it should be (the constraint variables $\tilde{x}$) and the easiest way of bringing about the required change (the suggested features $\hat{x}$). While in the first case features simply reflect the state of the world without reference to the sign, the behaviour of the other two aspects stems from interactions with constraints.

The simplest type of constraint we consider are referred to as atomic constraints. They are the basic units by which larger compound constraints are constructed. Therefore, they are also the smallest units on which advice generation is based. Any number of these constraints may be defined, each given a descriptive name. The constraint `higher-than` encountered in Figure 11.2 is such a constraint.

Atomic constraints serve three purposes. Their first purpose is to score a user's attempt. The constraint takes the input features relevant to it and provides a score ranging between zero (totally incorrect) and one (totally correct). This score represents the degree to which the user's attempt is satisfactory in the light of the specific constraint. A score of 0.5 is interpreted as being marginally correct. This aspect of the constraint is defined by a function $s_n(x)$, called its satisfaction function. This function maps relevant feature values to a score between zero and one as required by the constraint. In terms of fuzzy set theory (as formulated by Zadeh [86]), they are membership functions denoting the degree of membership of a parameter value to the set of satisfactory parameters.

The second purpose of atomic constraints is to actively constrain the constraint variables in $\tilde{x}$. While it is technically possible to simply assert the constraint $s(\tilde{x}) \geq 0.5$ (where s is the relevant satisfaction function), in the proof of concept system the legal values of $\tilde{x}$ are more explicitly defined. For the constraint `(higher-than stomach right-hand)` shown in Figure

11.2c, $(\tilde{y}_{RH} \geq \bar{y}_S)$ would be the explicit constraint. This is much more informative than the indirect constraint $s(\tilde{y}_{RH}) \geq 0.5$, easing the process of finding satisfactory values for $\tilde{y}_{RH}$.

The third purpose of atomic constraints is to generate advice after a set of suggested features has been obtained. Leaving aside for the moment the process of combining advice from different constraints, we first consider the case of generating advice based on a single constraint.

In order for a user to comprehend and correct their own error, they need at least two pieces of information:

- a description of the constraint that has been violated (for example, "The right hand should be above the stomach."); and
- instructions for correcting the error (for example, "Raise your right hand.").

The first type of feedback is absolute; it does not take into account the input features $\bar{x}$. As such, we refer to this type of advice as *static* advice. The advice does, however, take into account which elements of the input features $\bar{x}$ are depended on by the constraint. For example, in the constraint `(higher-than stomach right-hand)`, the static advice depends on the names of the body parts "stomach" and "right hand".

Static advice also depends on the state of inversion of the constraint. When the constraint is in its inverted form `(not (higher-than stomach right-hand))`, the advice must also be in inverted form. In the case of static advice, this simply means the advice itself is logically negated (for example, "The right hand should *not* be above the stomach.")

In the case of the second type of feedback, corrective feedback, the situation is more complex, as the type of feedback generated is dependent *additionally* on both the input features $\bar{x}$ and the recommended features $\hat{x}$. In isolation, the `higher-than` constraint, considered in the running example, might generate: "Raise your right hand." Because the advice depends on the user's current input and the suggested features, this type of advice is referred to as dynamic advice.

When constraints are combined, however, they may affect each other's dynamic advice. Consider adding the constraint `(lefter-than left-shoulder right-hand)`. Now the appropriate dynamic advice would be: "Move your right hand upwards and to the left."[1] This is why the suggested features $\hat{x}$ are needed. They provide a goal that has been reached by consensus between the constraints. Each atomic constraint then only needs to consider this set of suggested features when generating dynamic advice.

For the proof of concept system, constraint statement functions produced natural language feedback in the form a textual descriptions. Note that the exact nature of the advice could easily have been different. The most appropriate and accessible mode of advice can be chosen freely, whether this is natural language feedback, diagrams or animations. However, the system guarantees that the advice function will have both the current features $\bar{x}$ and a set of recommended

[1] While technically these constraints constrain independent features, generating a single piece of advice for position errors is preferable for the sake of clarity.

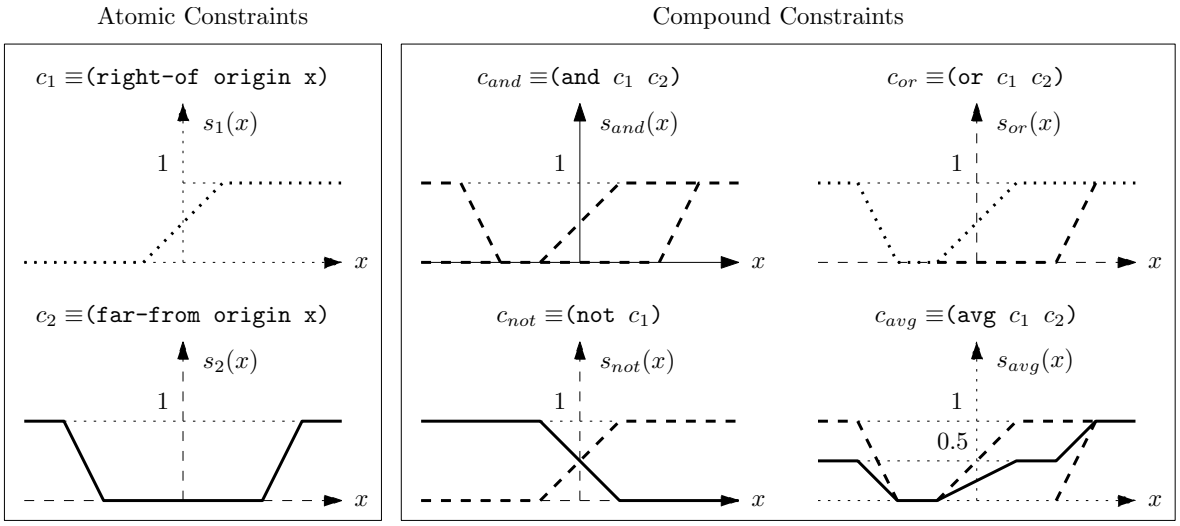


Figure 11.3: Examples of satisfaction functions for two atomic constraints `right-of` and `far-from` combined into compound constraints using the logical connectives 'and', 'or', 'not' and 'avg'. The satisfaction function s_{and} after the 'and' operation is the minimum of the two original satisfaction functions s_1 and s_2. Thus, everywhere either c_1 or c_2 is violated, c_{and} is violated, and c_{and} is only satisfied when both c_1 and c_2 are satisfied. The resulting constraint has the meaning "far to the right of". Similarly, after the 'or' operation, the satisfaction function s_{or} is the maximum of the two original satisfaction functions, so c_{or} is satisfied whenever any of the two constraints c_1 and c_2 are satisfied. The constraint c_{not} is the logical inversion of c_1, so the satisfaction function of constraint c_{not} is simply $s_{not} = 1 - s_1$. Wherever c_1 is satisfied, c_{not} is not satisfied, and vice versa. The constraint c_{avg} has a satisfaction function which is the average of the two original satisfaction functions. It can be seen that this connective gives partial credit for the satisfaction of both c_1 and c_2, weighted evenly.

features $\hat{x}$ to target. This removes the greatest burden when generating advice from several sources, allowing the individual atomic constraints to operate under the illusion of separation.

11.3 Compound constraints

It is exactly the challenge of obtaining a set of suggested least-effort consensus features $\hat{x}$ that is automated by the proof of concept system. It also automates the process of disentangling complex combinations of atomic constraints into a form which can be more readily understood by the user. We begin our discussion of these processes by defining how atomic constraints may be combined into richer compound constraints. As the internal structure of the compound constraints are known, this knowledge can be exploited to generate advice for the compound constraints automatically based on their constituent atomic constraints.

When combining constraints, the intended behaviour of the combined construct must be considered and the mathematical semantics must match this intention. Figure 11.3 illustrates the different operations discussed below.

If two constraints c_1 and c_2 must both be satisfied, the definition of the compound constraint $c_{and} \equiv \text{and}(c_1, c_2)$ is needed. A sensible approach would be to let c be violated to the same extent as the worst violated constraint of c_1 and c_2. This immediately suggests that the desired

satisfaction function should be $s_{and}(x) = \min(s_1(x), s_2(x))$, where s_1 is the satisfaction function of the first constraint c_1, and s_2 is the satisfaction function of the second constraint c_2.

Similarly, for the compound constraint $c_{or} \equiv \text{or}(c_1, c_2)$, let c be satisfied to the same extent as the most satisfied constraint of c_1 and c_2. The satisfaction function should then be $s_{or}(x) = \max(s_1(x), s_2(x))$.

Inverting a constraint $c_{not} \equiv \text{not}(c_1)$ should simply invert the satisfaction function $s(x) = 1 - s_1(x)$. A complete satisfaction of c_1 leads to a complete violation of c, and vice versa. A marginal satisfaction of c_1 is still a marginal satisfaction of c.

During sign segmentation, which will be discussed in Chapter 13, it is desirable to give partial credit for constraints that were satisfied by the user. The and-connective, however, gives no partial credit, being only as satisfied as the worst violated constraint. The avg-connective, which calculates the average of the satisfactions of its child constraints, can be used as a temporary substitute when partial credit has to be awarded. In the proof of concept system, it is only used internally and is not available in the domain specific language.

The logical connectives are as follows. Again, these are in line with Zadeh's original exposition of fuzzy set theory [86].

$$\begin{aligned}
\text{and}(c_1, \ldots, c_N) &\leftrightarrow \min(s_1(x), \ldots, s_N(x)) \\
\text{or}(c_1, \ldots, c_N) &\leftrightarrow \max(s_1(x), \ldots, s_N(x) \\
\text{not}(c_1) &\leftrightarrow 1 - s_1(x) \\
\text{avg}(c_1, \ldots, c_N) &\leftrightarrow \frac{1}{N} \sum_{n=1}^{N} s_n(x)
\end{aligned}$$

Here the notation $c \leftrightarrow s(x)$ means that the constraint c has the satisfaction function $s(x)$. For each c_n, we have $c_n \leftrightarrow s_n(x)$.

It is tempting to attach a probabilistic interpretation to the satisfaction function $s_n(x, y, \ldots)$, but ultimately this approach leads to semantic problems. If predicates constrain independent aspects of the features, then the way to combine the probabilities with and-semantics would be multiplication. However, this would mean that, even if individual constraint satisfaction probabilities were 0.99, adding more and more constraints would eventually drive their combined satisfaction probability below 0.5 (the satisfaction would be 0.99^n, with n being the number of independent constraints). This would clearly lead to pathological behaviour. Ultimately, it is not sources of uncertainty that are being combined, but degrees of correctness, which are known with certainty assuming the trustworthiness of the domain expert and given that the input features are known with sufficient accuracy.

Type	Example Constraint	Definition	Variables Constrained
Position	`(higher-than part other-part)`	Atomic	p_y of both parts
	`(at-height part other-part)`	Atomic	p_y of both parts
	`(close-to part other-part)`	Atomic	(p_x, p_y) of both parts
Velocity	`(moving-upward part)`	Atomic	v_y of the part
	`(moving-vertically part)`	Compound	v_y of the part
	`(moving part)`	Compound	(v_x, v_y) of the part
Orientation	`(palm-facing-left hand)`	Atomic	$\mathbf{o}_f$ of the hand
	`(palm-pointing-left hand)`	Atomic	$\mathbf{o}_p$ of the hand
Hand Shape	`(first-knuckle-unbent finger)`	Atomic	j_1 of finger
	`(finger-curled finger)`	Compound	All j_n of the finger
	`(hand-fist hand)`	Compound	j_n of all fingers

Table 11.1: Examples of constraints defined in the system, and the variables that they constrain. See Appendix B.1 for the full table. The terms `part` and `other-part` refer to named body parts (for example, the right hand, or the left shoulder). For the relevant body parts, p_x and p_y represent their positions, v_x and v_y their velocity, $\mathbf{o}_f$ and $\mathbf{o}_p$ the palm facing and pointing vectors (which determine the hand orientation), and j_n, the finger joint angles.

11.4 Constraint types

Table 11.1 summarizes the types of constraints that the proof of concept system implements (see Appendix B.1 for the full table), providing a subset of the available constraints in each case as examples. These include constraints on hand position, velocity, orientation and shape. Note that these correspond exactly with the types of phonetic elements that are important in sign languages.

The only phoneme types not considered are those of non-manual gestures. While these form an important part of sign language words, extending the system to include constraints on non-manual gestures would be simple, given a body tracker which can provide the features needed by these new constraints.

11.5 Summary

In this chapter, the basic components of the tutor were defined, namely features and atomic constraints. It was also shown how compound constraints may be built from atomic constraints using logical connectives. The process of generating advice given such constraints and input features is discussed in the next chapter.

Chapter 12

Single-frame Advice Generation

The generation of advice for atomic constraints was discussed in the previous chapter. Each atomic constraint is able to generate static and dynamic advice given the input features $\bar{x}$ and a set of suggested features $\hat{x}$. When atomic constraints are combined into compound constraints, and these compound constraints are combined into even higher-level constraints, the generation of advice is no longer straightforward.

There exist a number of desirable qualities for advice generated from a combination of multiple constraints. These include:

- individual items of advice must be correct;
- the specificity of advice generated from atomic constraints needs to be retained when combined at a higher level;
- combination of individual items of advice must not lead to paradoxes; and
- the logical structure of the resulting advice needs to be in a form that a typical user will be able to understand. Ideally, the advice should take the form of a simple to-do list, where the user simply has to follow each suggestion in turn.

It is useful, at this stage, to represent compound constraints in the form of trees. Figure 12.1 shows how the logical connectives can be represented as trees. Each connective is the root of a tree with the subconstraints being children of the root node.

Figure 12.2 gives an example of a more complex tree of constraints. This example demonstrates a case where disentangling the structure of the compound constraint in order to generate advice is more challenging. The process used by the proof of concept system will be discussed in the following sections, using this tree as a running example. Each step of this process is visualised in Figure 12.2.

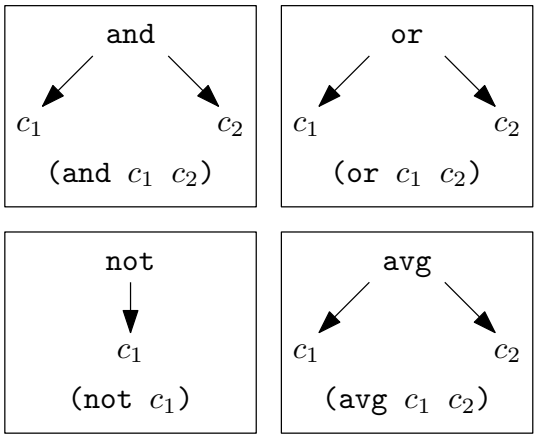


Figure 12.1: The basic logical connectives and their corresponding representations as elementary constraint trees.

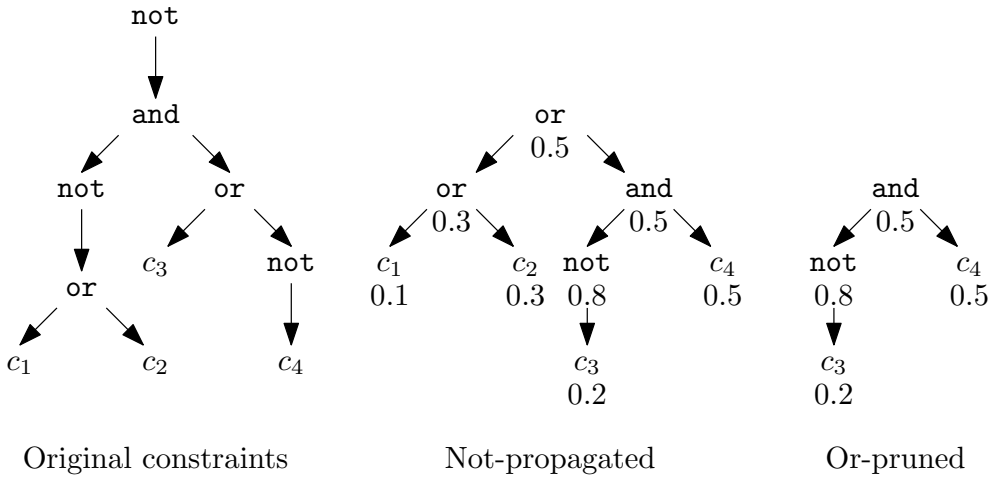


Figure 12.2: The constraint tree representing `(not (and (not (or` c_1 c_2`)) (or` c_3 `(not` c_4`))))` at different stages during the advice generation process. The leftmost tree is the original constraint tree. This is followed by the tree after not-propagation. Note that no high-level not-connectives remain, with the only remaining not-connectives directly inverting atomic constraints. After not-propagation, each subtree is labelled with its satisfaction based on the suggested features $\hat{x}$. Or-pruning is then performed, meaning each or-operation is replaced by its most satisfied subtree. The rightmost tree is the result of this operation.

12.1 Not-propagation

Perhaps the greatest obstacle to generating easily comprehended feedback from a compound constraint tree is the prospect of nested not-inversions of complex compound predicates. If a direct approach is taken, a user would have to keep track of the state of inversion in the current subtree they are trying to "parse" mentally.

The example in Figure 12.2 illustrates this situation. As can be seen, the multiple levels of negation in the leftmost tree seem to introduce a level of complexity not easily converted into unconvoluted advice.

However, it can be shown that De Morgan's laws

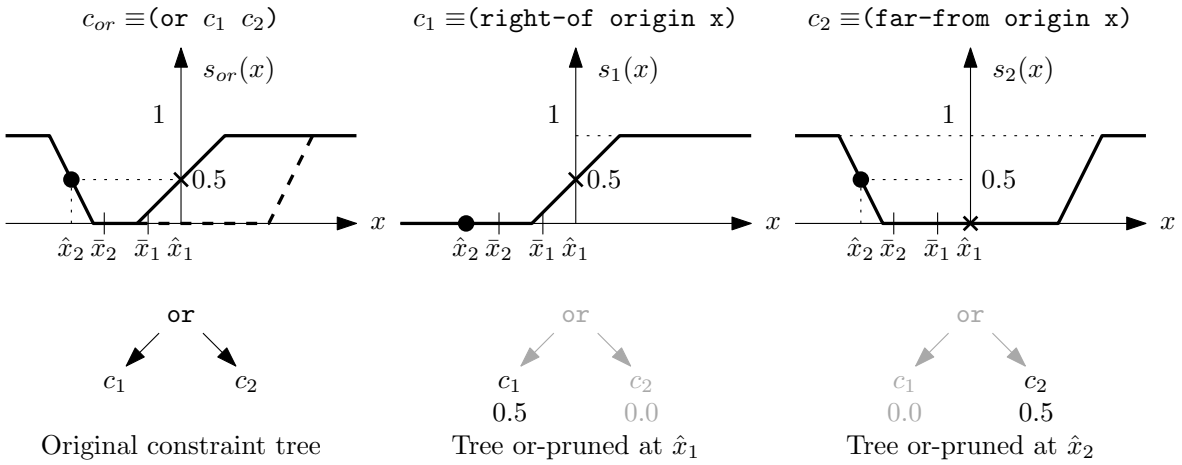


Figure 12.3: A basic example of or-pruning. The compound constraint on the left c_{or} is the result of or-ing the two constraints c_1 and c_2. Two possible values for the input features are considered. The first possible input value $\bar{x}_1$ is associated with the suggested features $\hat{x}_1$ to its right. Or-pruning at $\hat{x}_1$ leaves only the constraint c_1, for which the advice "Move x to the right" would be generated. The second input value $\bar{x}_2$ is associated with the suggested features $\hat{x}_2$ to the left. Or-pruning at $\hat{x}_2$ leaves only the constraint c_2, for which the advice "Move x to the left" would be generated. Note that, in both cases, the constraints would generate advice in isolation which would be contradictory when combined. In this way, or-pruning eliminates one potential source of paradoxical advice.

$$\begin{aligned} \text{not}(\text{and}(c_1,\ldots,c_N)) &= \text{or}(\text{not}(c_1),\ldots,\text{not}(c_N)) \\ \text{not}(\text{or}(c_1,\ldots,c_N)) &= \text{and}(\text{not}(c_1),\ldots,\text{not}(c_N)) \end{aligned}$$

hold for these fuzzy connectives [86].

In combination with double-negation $\text{not}(\text{not}(c)) = c$, these properties can be used to "propagate" all not-operators downward in the tree. The result of this operation on the example tree in Figure 12.2 is shown immediately to the right of the original constraint tree.

Note that, after not-propagation, the only inversions are inversions of atomic constraints, in this case, $\text{not}(c_3)$. As noted earlier, generation of advice from atomic constraints which have been inverted, is relatively simple. As high-level not-operations are no longer present, we proceed to the next stage of the advice generation process, namely or-pruning.

12.2 Or-pruning

A not-propagated tree is some combination of and- and or-operations nested in various ways, with uninverted or directly inverted atomic constraints at the leaves. The central tree in Figure 12.2 reflects this situation. The means by which advice can be generated from this structure are still not clear. It is, however, possible to remove the or-operations in a process we will refer to as or-pruning.

Or-pruning relies on the observation that, for any given value of the features, one of the subconstraints of an or-operation will be more satisfied than the other(s). We can then simply

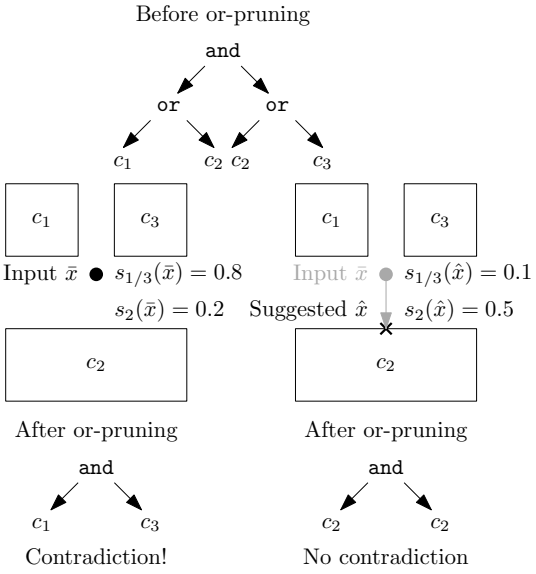


Figure 12.4: Preventing contradictions due to or-pruning. In this example, the satisfaction values for the constraints c_1, c_2 and c_3 evaluated at the input features $\bar{x}$ are set to $s_1(\bar{x}) = 0.8$, $s_2(\bar{x}) = 0.2$ and $s_3(\bar{x}) = 0.8$ respectively. Evaluated at the suggested features $\hat{x}$, the satisfaction values are defined as $s_1(\hat{x}) = 0.1$, $s_2(\hat{x}) = 0.5$ and $s_3(\hat{x}) = 0.1$ respectively. Or-pruning at the input features leads to a contradiction, whereas or-pruning at the suggested features avoids this paradox.

discard all but the most satisfied subconstraint, as the others are less attractive alternatives made superfluous by the most satisfied subconstraint. Ties may be broken by arbitrarily choosing one of the subconstraints. As will be explained shortly, the constraints must be evaluated for the suggested features $\hat{x}$, and not the input features $\bar{x}$.

Figure 12.3 gives an example of this operation. Note that this also eliminates one possible source of paradoxical advice, as or-operations often merge the satisfied regions of subconstraints which are entirely mutually exclusive, and so cannot be satisfied at the same time.

Or-pruning of the running example is shown in Figure 12.2, where or-pruning transforms the central tree to the rightmost tree. Each constraint is labelled with its level of satisfaction for the suggested features $\hat{x}$, followed by the replacement of each or-connective by its maximal subtree.

At first glance, it seems appropriate to do or-pruning by simply using the input features $\bar{x}$ to evaluate each constraint. It would certainly be a simple operation, as the constraints within the not-propagated tree are easily evaluated.

Upon reflection, it becomes clear that this is a further potential source of paradoxical advice. Consider Figure 12.4. Here a constraint tree is pictured before or-pruning and after or-pruning. The domain of satisfactory points for each subconstraint is indicated as rectangular regions.

On the left hand side, we simply use the input features $\bar{x}$ to do or-pruning. In each subtree, the least violated constraint is chosen (c_1 and c_3 respectively), and retained. However, this leads to a paradox. The domain of satisfactory points of c_1 and c_3 do not overlap, so it is impossible to satisfy both c_1 and c_3 at the same time.

Now, if we instead find a nearby point to the input features in the feature space where the

constraint tree has a satisfaction of at least 0.5, we can use this point to do or-pruning instead. This point, the suggested feature point, is in the domain of satisfactory points for c_2. Or-pruning now leads to the tautology `(and` c_2 c_2`)`.

While this approach prevents paradoxical advice, it makes the advice generation process significantly more complicated. Instead of merely evaluating the constraints at $\bar{x}$, there are a host of possible $\hat{x}$'s to consider. However, in order to immediately discuss how the results of or-pruning are actually used, the process of obtaining the suggested features $\hat{x}$ is deferred to Section 12.4.

12.3 And-prioritisation

For any tree of constraints, the result of not-propagation followed by or-pruning will be a set of atomic constraints with possible inversion. These are connected using possibly nested and-operations. It is easy to replace this tree with a list of possibly inverted atomic constraints, joined by a single and connective. Advice may then be generated for each of them, emphasising the advice of the currently most violated constraints given the input features $\bar{x}$.

This list of advice is exactly the simple to-do list that was desired at the start of advice generation. Even though the list might be long, the user at least knows that each piece of advice can be followed together unconditionally. The advice is also automatically prioritised for the user by virtue of its position in the list.

The final list of constraints for the running example is c_4 and not(c_3). If the satisfaction for not(c_3) based on the input features is 0.3 and the satisfaction for c_4 is 0.4, then the and-prioritised list, with the highest priority first, would be `(not` c_3`)` and c_4.

It should be noted that, even for constraints which have a satisfaction of more than 0.5, advice may need to be generated from the constraint. While the constraint may have been satisfied for the input features, the process of satisfying the other constraints may have indirectly caused a change in the features on which the satisfied constraint depends. Figure 12.5 illustrates such a situation.

There are at least two approaches one can take to deal with this problem. The first approach is to generate advice for all the constraints remaining after and-prioritisation. However, if there is no change in the features on which a constraint depends, then no advice needs to be generated based on the constraint. One may also require a significant change in these features, based on some measure of significance.

Another approach, which is adopted here, is to ignore advice for already satisfied constraints, and instead try to avoid situations where an atomic constraint has regions of satisfaction which are different in meaning. This is illustrated in (b) of Figure 12.5, where the atomic constraint c_2 is replaced by a compound constraint which is the or of two subconstraints c_{21} and c_{22}. A practical example of this is the `moving-vertically` constraint, which constrains a body

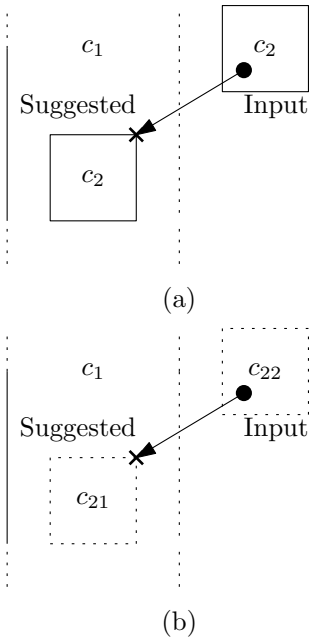


Figure 12.5: Already satisfied constraints may still need advice. (a) Here the constraint tree is `(and` c_1 c_2`)`. c_1 is violated by the input, but c_2 is not. The suggested point is in an entirely different satisfied region of c_2, and so advice may have to be generated based on c_2, even though it is already satisfied. Note that c_1 depends only on the horizontal axis variable, but c_2 depends on both axis variables, so the only way we know advice must be generated for the vertical axis variable is through c_2. (b) The previous situation may also be resolved heuristically by splitting c_2 into two subconstraints $c_2 \equiv$ (`or` c_{21} c_{22}).

part to be moving at a significant speed either upward or downward. It is defined as a compound constraint which is the or of the constraints `moving-upward` and `moving-downward`.

It should be stressed that this is a heuristic solution. However, this approach has the advantage of avoiding "spamming" the user with constraints that are likely to stay satisfied once the violated ones are dealt with. Even if this is not the case, it is possible that the user may still converge to the correct behaviour after a few attempts. If subsequent user tests show that this heuristic is lacking, it would be easy to return to the strategy of generating advice for all constraints left after or-pruning.

12.4 Suggested features

Having established the need for a set of suggested features, a means by which they can be found will now be presented. Systems implementing this operation must:

- allow candidate features with domains of sufficiently fine granularity so that any meaningful differences between candidates are retained;
- allow constraints to be placed on these features, and these constraints must be sufficiently expressive so that constraints of very different character may be defined;

- allow the definition of some measure of "effort" $E(\bar{x},x')$ between input features $\bar{x}$ and candidate features x', so we may determine which suggested features would lead to the least effort expended by the user (E will be referred to as an energy function); and
- Allow the effort measure to be minimised while at the same time satisfying the constraints.

The system's implementation makes use of feature variables which are constrained to be integers within a finite interval (for example, all integers from 4 to 15). This seems like a rather strong assumption, but this concern is easily set aside when it is realised that this allows the use of fixed-point arithmetic to simulate real-valued variables. Quantities like the position of hands and different body parts are discretised with controllable levels of accuracy.

As will be discussed shortly, this assumption also forms a key part of turning the simultaneous optimization and constraint satisfaction problem into a series of pure constraint satisfaction problems. This allows us to define constraints with less need to consider their relationship to the optimisation procedure.

The procedure presented here performs a global optimisation. Therefore, the results presented in later chapters are based on the best possible solutions available to the system.

12.4.1 Equivalent constraints

If the membership function of a constraint is $s(x)$, then by definition the constraint $s(x) \geq 0.5$ will implement the desired system behaviour. However, it is not necessarily convenient to define constraints in terms of their true membership functions. Instead, defining the constraint in terms of the decision boundaries *implied* by $s(x)$ is generally much more convenient for constraint satisfaction procedures. So, for example, the constraint in Figure 11.2c might be rendered as $y \geq \bar{y}_S$ instead. This is a much more informative constraint than the indirect $s(y) \geq 0.5$.

All constraints in the system therefore define *both* a membership function for use in scoring and an equivalent constraint function for use in constraint solving. The constraint satisfaction facilities of the Screamer [61] package were used to solve the pure constraint satisfaction problems. The package allows the assertion of a variety of constraints including arithmetic relationships, equalities and inequalities. Importantly, constraints may be combined using `and`, `or` and `not` semantics, and so forming compound constraints is easy (given that well-behaved versions of the atomic constraints have been defined).

12.4.2 Energy functions

As discussed previously, an energy function $E(\bar{x},x')$ needs to be defined to quantify the effort expended by the user in correcting their attempt by changing the current input features $\bar{x}$ to x'. We wish to find the x' which minimises $E(\bar{x},x')$ while satisfying the constraint problem. This x' is both a correct solution and a minimum effort solution, and becomes the suggested features $\hat{x}$.

In general, constrained optimisation (that is, solving both the constraint and optimisation problems at once) is more difficult than constraint satisfaction without optimisation. However, as described in the next section, we can transform the constrained optimisation problem into a series of constraint satisfaction problems if we have the following:

- It must be possible to add the expression $E_m \leq E(\bar{x}, x') \leq E_n$ (where E_m and E_n are known integers with $E_m \leq E_n$) as a constraint to the existing constraint satisfaction problem. Using this added constraint, it becomes possible to test whether a solution to the constraint problem exists where the expended effort is within the range from E_m to E_n. By testing increasingly restricted energy intervals, it becomes possible to find solutions to the constraint problem of given energies. In particular, restricting the search interval as in a binary search allows one to find a minimum energy solution, as detailed in the next section.

- The energy function $E(\bar{x}, x')$ must be non-negative, integer valued, and bounded above over the allowable domain of the feature variables. Therefore, the range of the energy function is restricted to a finite interval, with a finite number of possible values.

Together, these ensure that continually dividing intervals on the energy range into non-empty subintervals will eventually terminate in intervals containing a single energy value. This guarantees that search by subdivision of the allowed energy range will terminate.

12.4.3 Optimisation as binary search

We now outline a branch and bound strategy used to find suggested values in the system, emphasising that $E(\bar{x}, x')$ *must* be integer-valued for the algorithm to work.

To find $\hat{x}$, we start by finding *any* solution x'_0 to the original constraint problem. This represents an upper limit for the minimum of the energy $E^+_{min} = E(\bar{x}, x'_0)$. This step has the added utility of determining whether there, in fact, exist solutions to the constraint problem at all (if not, the definition of the sign interval is faulty). Because the energy is non-negative, we know that $E^-_{min} = 0$ is a lowest possible minimum energy. The minimum energy must then be somewhere in the interval $E^-_{min} \leq E_{min} \leq E^+_{min}$, and we may determine this minimum energy by means of a binary search (with slight modifications).

For each refinement step, if $E^+_{min} = E^-_{min}$ we are done and the last solution to the constraint problem was in fact also a minimum energy solution. If not, we find the energy midway between the upper and lower bound $E'_{min} = \left\lfloor \frac{E^-_{min} + E^+_{min}}{2} \right\rfloor$. We now solve the original constraint problem with the added constraint $E(\bar{x}, x') \leq E'_{min}$. If a solution x' is found, then set $E^+_{min} = E(\bar{x}, x')$. This is better than simply setting $E^+_{min} = E'_{min}$ because a new upper bound for the minimum energy has been found. If no solution is found, set $E^-_{min} = E'_{min} + 1$, because the minimum energy can no longer be less than or equal to E'_{min}.

This process has the virtue of guaranteed termination (assuming the embedded constraint solver always terminates) and the optimisation is solved using $O(\log(E^{+}_{min}))$ constraint problems.

12.5 Feature representation

During the preceding discussion, it was stressed that the energy function must be integer valued, and that the feature variables are also quantised with a finite domain. It is necessary to comment more exactly on these representations, and how they are related to the global energy function. This is particularly true in the case of hand orientation.

Representing positions and velocities are relatively simple. The positions of various body parts are represented in quanta which are some fixed fraction of the width of the head. Velocities of body parts are represented in quanta which are some fixed fraction of one head width per second. The joint angles of the fingers are represented in degrees, and quantised in 10 degree steps.

The global error function in the proof of concept system is the sum of a set of energy functions associated with different subsets of the complete set of features. For example, the position of the right hand contributes the square of the Euclidean distance between the input features and the candidate features $(\tilde{x}_{RH} - \bar{x}_{RH})^2 + (\tilde{y}_{RH} - \bar{y}_{RH})^2$ to the global energy function. Because the position features are integer valued, the energy they contribute is also integer valued, and the condition is automatically satisfied.

The same type of error function is used for the velocities of the hands, and the joint angles of the hands. It was found to be unnecessary to weight each of the energies to establish relative importance, though this possibility remains open.

12.5.1 Representing orientation

The situation is more complex when examining hand orientation. This is because there are competing considerations at play.

Formally, the choice is between the possible representations of the three-dimensional rotation which embodies a hand's orientation. Possible representations include quaternions, Euler angles, rotation matrices and axis-angle representations.

In making a choice between orientation representations, there are several key considerations. Certain types of constraints are "natural" given a representation in that the constraints are simple to express in the constraint satisfaction system. It must also be possible to define appropriate energy functions easily.

Furthermore, the kind of constraints that are natural given a representation must also be natural from a phonological perspective, in that they directly correspond to the type of constraints that typically form part of phonemes. It must also be simple to generate advice from these constraints that can be readily understood by the user.

Ultimately, a representation entirely in terms of vectors was chosen. In particular, hand orientation is represented using the vector $\mathbf{r}_f$ normal to the inside of palm, and the vector $\mathbf{r}_p$ tangential to the palm pointing in the direction of the non-thumb fingers when outstretched. The vectors $\mathbf{r}_f$ and $\mathbf{r}_p$ are referred to as the facing vector and the pointing vector respectively. There are several reasons why this representation is desirable.

Firstly, signs often include pointing at regions of space, with pronouns being an example of this. Thus, being able to explicitly specify the pointing vector is phonologically convenient.

Secondly, the pointing vector does not uniquely specify a rotation. Two possible ways to resolve this are the addition of a rotation angle around the pointing vector, or the addition of the facing vector.

Finally, the addition of the facing vector was chosen because of the need to generate advice that can be easily understood. Consider generating advice when a rotation angle is used instead. For a rotation angle to have meaning, it must be defined relative to an axis, which would be some vector normal to the pointing vector. The angle then represents a choice from the set of vectors normal to the pointing vector, with the reference vector chosen if the angle equals zero.

Note that this means the user has to perform two mental operations. They need to understand where the reference vector is pointing, and so need to understand the mapping of pointing vectors to reference vectors (which is as yet not specified). Then they must perform a mental rotation of the reference vector based on the pointing vector and the rotation angle.

If the direction of the facing vector is simply described, no additional mental operations are needed, as the rotation is now uniquely specified, and so is particularly easy to comprehend.

In addition, the pointing- and facing-vector representation has useful mathematical properties. It is possible to specify constraints of interest by simply constraining the dot product of these vectors with themselves, with each other, or with constant valued reference vectors. This avoids the need for including trigonometric functions when constraining the feature variables, or in the calculation of the associated energy.

The representation does include redundant information, as it has six degrees of freedom, while only three degrees of freedom are needed to specify a three-dimensional rotation (using Euler angles for example). Constraining their lengths to equal one and their dot product to equal zero (so that they are perpendicular) removes these additional degrees of freedom.

Note that the vectors must be represented using integer values. Therefore, each vector is multiplied by some sufficiently large integer N and its components rounded to the nearest integers, forming $\mathbf{r}'_f = \text{round}(N\mathbf{r}_f)$ and $\mathbf{r}'_p = \text{round}(N\mathbf{r}_p)$. The fact that their lengths change has to be factored into all subsequent constraints, but these adjustments are simple. Because the values are quantised, this does introduce the need to express the unit length constraints of $\mathbf{r}_p$ and $\mathbf{r}_f$ as the approximating inequalities

$$N^2 - \delta_l \leq \mathbf{r}'_f \cdot \mathbf{r}'_f \leq N^2 + \delta_l$$

$$N^2 - \delta_l \leq \mathbf{r}'_p \cdot \mathbf{r}'_p \leq N^2 + \delta_l$$

where δ_l is some integer which specifies the allowed error in the lengths of the vectors. The vectors $\mathbf{r}_f$ and $\mathbf{r}_p$ can be similarly constrained to be approximately perpendicular using

$$-\delta_p \leq \mathbf{r}'_f \cdot \mathbf{r}'_p \leq \delta_p$$

where δ_p is some integer which specifies the allowed error in the dot product relative to zero.

12.5.2 Constraining orientation

It now becomes possible to represent a constraint on either the pointing- or facing-vector by employing the dot product of one of these vectors with a reference unit vector ϕ using the well-known relationship

$$\cos(\theta) = \mathbf{r} \cdot \phi$$

where $\mathbf{r}$ is either the facing- or pointing-vector, and θ is the angle the vector $\mathbf{r}$ makes with the vector ϕ. Therefore, θ acts as an indication of how well aligned the two vectors are. If θ is zero, $\cos\theta = 1$ and they are co-linear. If θ is $\frac{\pi}{2}$, $\cos\theta = 0$ and they are perpendicular. If θ is π, $\cos\theta = -1$ and they are oriented in opposite directions.

Therefore, we can render a constraint on the direction of the vector $\mathbf{r}$ using the inequality

$$\cos(\theta_c) = \lambda \leq \mathbf{r} \cdot \phi.$$

where θ_c is the maximum allowable angle that the orientation vector $\mathbf{r}$ may make with the reference direction ϕ. Here, λ is used to emphasise the fact that $\cos\theta_c$ is a constant, and can be precalculated. Therefore, our representation is free of trigonometric functions. This is particularly important, as our features of interest are restricted to integer values (possibly representing fixed point numbers).

The constraint `(palm-pointing-upward hand)` may hence be rendered as

$$\cos\left(\frac{\pi}{4}\right) = \frac{1}{\sqrt{2}} \leq \mathbf{r}_p \cdot \mathbf{u}_y$$

where $\mathbf{u}_y$ is the unit vector pointing upwards. Here, we have defined an upward pointing vector as any pointing vector that makes at most an angle of $\frac{\pi}{4}$ with the vector pointing upwards.

Satisfaction functions for this type of constraint can also be defined in terms of the dot product, so that the previous constraint would have the form

$$s(\mathbf{r}_p) = s'(\mathbf{r}_p \cdot \mathbf{u}_y) = s''(\theta).$$

This has the same form as the types of satisfaction functions already defined, and so an appropriate piecewise-linear satisfaction function may be defined for $s''(\theta)$. The variable being constrained is the error in the direction of the pointing vector measured in radians. For this particular example, $s''(\theta) \geq 0.5$ for all $\theta \leq \frac{\pi}{4}$, and $s''(\theta) \leq 0.5$ for all $\theta \geq \frac{\pi}{4}$.

12.5.3 Orientation energy functions

The energy of a candidate orientation is also simple to define using the pointing- and facing-representation. We can define a separate energy function for each of these vectors. If $\bar{\mathbf{r}}$ is the input value of the particular orientation vector, and $\tilde{\mathbf{r}}$ are the constraint variables associated with that vector, then the square of the Euclidean distance between these vectors

$$E_{\mathbf{r}} = ||\mathbf{r} - \tilde{\mathbf{r}}||^2$$

in three-dimensional space can be used as an energy function.

12.5.4 Orientation advice

The ease of generating advice based on orientation constraints depends on the mode of feedback. The advice generation facility of the constraint will have both the current and suggested orientation vector. The possibilities for natural language feedback seem rather limited, essentially amounting to restating the static advice. For example, the static advice "Your right hand should be pointing upward." would be accompanied by the dynamic advice "Rotate your right hand until it is pointing upwards."

However, this feedback hides the information available to the advice generation system. A more natural mode of feedback would be an animation between the current hand orientation and the suggested orientation. Implementing such an animation would be relatively simple with the given information. The animation may then be accompanied by the natural language feedback, which makes clear the motivation behind the animation.

Note that this would be unlike the synchronized template animations available in some existing tutors [58, 91]. The suggested orientation is that orientation which is easiest to assume given the current hand orientation, while synchronized template animations[1] merely show the "best" possible hand orientation according to the template, and therefore do not take user effort into account.

12.6 Summary

This chapter presented the process by which advice is generated when given a set of features associated with a single frame of video, and the constraints that the user must obey at that

[1]The synchronisation operation at least attempts to find the closest "best" value available *over time*.

moment in time. It has been shown that not-propagation, or-pruning and and-prioritisation make it possible to define complex constraints on the movements, while still being able to represent the final advice in a form which is a simple conjunction of individual items of advice.

Because a procedure for obtaining a set of suggested features has been provided, each constraint may be used to generate advice under the illusion of isolation, because the suggested features are guaranteed to satisfy all the constraints at the same time. In addition, the procedure also minimises a measure of the user's effort.

In the following chapter, the generation of advice based on entire video sequences is described, where the single-frame processes described in this chapter are used as the fundamental subcomponent.

Chapter 13

Video-level Advice

In the preceding chapter, it was assumed that constraints were all applied at a single moment in time to a set of features obtained from a single frame of video. In this chapter, the generation of advice for entire video sequences will be discussed. At this level, single-frame advice generation is used as a subcomponent.

13.1 Sign specification

If single-frame advice generation is to be the subcomponent used to generate video-level advice, then the central aspects of generating advice at sign level become: (1) the selection of the individual video frames for which to generate advice and (2) the set of constraints against which those video frames will be evaluated.

The selection of the individual video frames is performed during *sign segmentation*, the process by which the video is divided into the different phases of the sign. For example, the sign "woman" has three phases; a frame of video from each was shown in Figure 11.1.

In order to do sign segmentation, a way of defining each phase of the sign is needed. A natural way of describing a sign would be to use a compound constraint to describe each phase of the sign. For the sign "woman", the first phase of the sign may be described by the compound constraint

```
(and
  (unmoving right-hand)
  (at-height left-chest right-hand)
  (same-x left-chest right-hand)
  (hand-pose-woman right-hand))
```

The and-connective is used to build this compound constraint by combining the atomic constraints `unmoving`, `at-height` and `same-x`, as well `hand-pose-woman`, a named compound constraint. Named compound constraints will be discussed in Section 13.4.3.

```
(defsign woman
  (start-when
    (or (higher-than stomach right-hand)
        (at-height stomach right-hand))
    (unmoving right-hand))
  (interval ()
    (unmoving right-hand)
    (at-height left-chest right-hand)
    (same-x left-chest right-hand)
    (hand-pose-woman right-hand))
  (interval ()
    (not (moving-vertically right-hand))
    (moving-rightward right-hand)
    (at-height left-chest right-hand)
    (hand-pose-woman right-hand))
  (interval ()
    (unmoving right-hand)
    (at-height right-chest right-hand)
    (same-x right-chest right-hand)
    (hand-pose-woman right-hand))
  (stop-when
    (moving-downward right-hand)
    (lower-than stomach right-hand)))
```

Figure 13.1: Domain specific language definition of the SASL sign "woman".

The constraint captures the fact that, during this phase of the sign, the right hand must be still, at the position of the left chest, and with the hand pose defined within `hand-pose-woman`.

As a sign is composed of different phases, a way of specifying several such phases is needed. Figure 13.1 illustrates how this is done in the system. The phases of the sign are specified inside the clauses marked `interval` (the `start-when` and `stop-when` clauses will be discussed in Section 13.2). The constraint from the first phase is the first `interval` clause. The second and third phase of the sign follow afterwards. Note however that the and-connective has disappeared from the constraints. That is because the `interval` clauses are treated as implicit and-operations.

Intervals have the following general form:

```
("interval" parameters
  compound-constraint
  compound-constraint
  ...)
```

In the example code for `woman`, there were no parameters. Parameters will be discussed in Section 13.4.1.

Note that the constraints are specified from the perspective of the *signer*, so the constraint `moving-rightward` is satisfied by the relevant body part going leftward in the video sequence. Furthermore, the sign definitions are biased towards right-handed signing. Left-handed signs can always be obtained by performing the appropriate transformations on the constraints.

13.2 Sign segmentation

When the system is presented with a sequence of video frames, it is first necessary to determine which frames of video belong to which intervals of the sign. This process is called sign segmentation.

During sign segmentation, a segmentation score is calculated for each interval at each frame. Figure 13.2 shows an example of segmentation scores calculated for each interval of the sign "woman" for an incorrect signing attempt. At each given moment between frames 52 and 150, one of the three intervals scores higher than the other intervals. This tendency is used to decide which interval is active during any given moment in time.

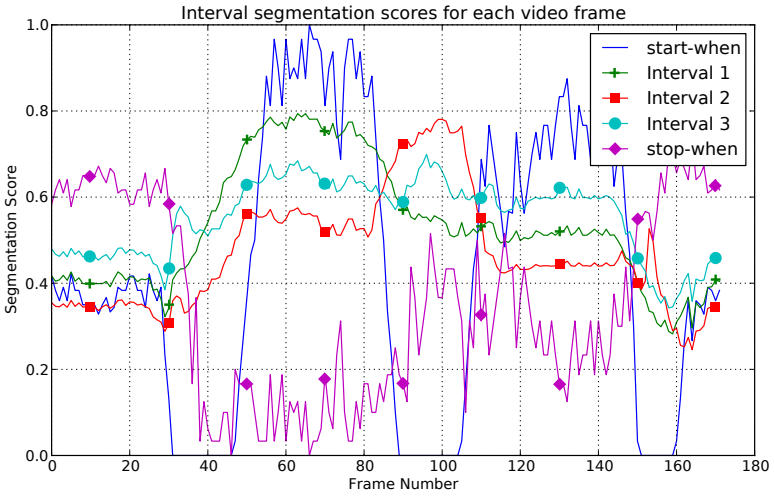
Clearly the meaningful portion of a sign is limited in extent for a given video. In Figure 13.2, this is the interval from frame 52 to frame 150. In order to retain only the meaningful portion of the sign, the domain specific language includes the `start-when` and `stop-when` clauses. These have the general form

```
("start-when"/"stop-when"
  compound-constraint
  compound-constraint
  ...)
```

The compound constraints in these clauses are combined using and-semantics. The meaningful portion of a sign starts when a satisfaction of at least 0.5 is calculated for the `start-when` clause (or the first frame if no start-when clause is present). The sequence ends at the frame after which the `stop-when` clause has a satisfaction of at least 0.5 for the rest of the video (a search for the first satisfaction below 0.5 is performed from the end of the video). If no `stop-when` clause is present, the sequence ends at the last frame.

In Figure 13.2, the first 51 frames were skipped as the `start-when` clause first achieves a score of 0.5 at frame 52. The frames after frame 150 are skipped because the `stop-when` clause scores 0.5 for all frames after that.

For interval clauses the segmentation scores are not the satisfactions of the compound constraints, but are a heuristically calculated alternative that gives partial credit for the extent to which each subconstraint is satisfied. This is done by replacing all and-connectives with avg-connectives. The first interval of "woman" becomes



(a)

(b)

Figure 13.2: (a) Segmentation scores for each interval (and satisfaction of `start-when` and `stop-when` clauses) for an incorrect attempt of the SASL sign "woman" (the hand is too low). Note that, while part of the sign is done incorrectly, the segmentation scores are still usable to correctly segment the sign. The segmentation results placed interval 1 between frame 52 and 86, interval 2 between frame 86 and 109, and interval 3 between frame 109 and 150. The maximum segmentation score frames for each interval were frame 63, frame 99 and frame 112 respectively. These three maximum frames are shown in (b) in that order. Advice for each interval is based on these maximum segmentation score frames. The correct advice was generated in all three frames (the static advice "Your right hand must be at the same height as your left chest." and the dynamic advice "Move your right hand upwards.").

```
(avg
  (unmoving right-hand)
  (at-height left-chest right-hand)
  (same-x left-chest right-hand)
  (hand-pose-woman right-hand))
```

The named compound constraint `hand-pose-woman` is expanded, and all and-connectives internal to it are also converted to avg-connectives.

Now, instead of just retaining the lowest satisfaction of all the child constraints, the avg-connective takes the average of the satisfactions. This preserves information from each of the constraints. This is especially important when it is realised that there is an *expectation* that users will perform signs incorrectly, which has a tendency to drive compound constraint satisfaction

close to zero. This is because of the relative harshness of the and-connective. The or- and not-connectives, however, are left unchanged.

Viterbi alignment [51] is used to determine the final assignment of frames to intervals. If $\mathbf{q} = (q_1, q_2, \ldots, q_N)$ represents a state sequence, where q_n is the chosen sign interval number at video frame number n, the alignment is obtained by maximization over all the possible frame assignments

$$\mathbf{q}_{\max} = \underset{\mathbf{q}}{\operatorname{argmax}} \left[\pi_{q_1} + \omega_{q_N} + s'_{q_1}(\bar{x}_n) + \sum_{n=2}^{N} \left(a_{q_{n-1}q_n} + s'_{q_n}(\bar{x}_n) \right) \right].$$

Here s'_i is the segmentation score of the constraint associated with interval i, with the avg-connective replacing and-connectives as described previously. The score s'_i is pseudo-probabilistic in that it assumes the usual role of the logarithm of the observation probability in an HMM, but is itself not a probability.

The input features of the video frame are represented by $\bar{x}_n$. The term π_{q_1} is the pseudo-log-probability of the sequence starting in the state q_1. By default, this is $\pi_1 = 0$ (allowed) and $\pi_n \to -\infty$ (disallowed) otherwise. This means that, by default, a sign will always start with the first interval. The term ω_{q_N} is the pseudo-log-probability of the sequence ending in the state q_N. By default, this is $\omega_N = 0$ (allowed) and $\omega_n \to -\infty$ (disallowed) otherwise. This means that, by default, a sign will always end with the final interval. The term $a_{q_{n-1}q_n}$ is the pseudo-log-probability associated with transitioning from state q_{n-1} to state q_n. By default, frames from an interval may be followed by more frames of the same interval, or frames from the next interval. This means that $a_{nn} = 0$ (allowed), $a_{n(n+1)} = 0$ (allowed) and $a_{mn} \to -\infty$ (disallowed) in all other cases.

The default behaviour of the preceding parameters may be changed using interval parameters and transition clauses, as described in Section 13.4.1.

For the attempt in Figure 13.2, based on the scores in Figure 13.2a, the input alignment procedure placed interval 1 between frame 52 and 86, interval 2 between frame 86 and 109, and interval 3 between frame 109 and 150. In each case, this roughly corresponds with the regions where each particular interval scores the highest.

13.3 Frame selection and advice generation

After each interval has been assigned a sequence of frames, advice may be generated for each of these intervals. Rather than attempt to generate advice over the entire interval, a single representative frame is chosen. This is done heuristically by selecting the frame with the highest segmentation score. In the example, the maximum segmentation score frames for each interval were frame 63, frame 99 and frame 112 respectively. These are also the frames shown in Figure 13.2b. Intervals are treated separately, and advice is generated for each interval based on the frames identified during sign segmentation and frame selection, as described in Chapter 12.

13.4 Additional language constructs

In this section, a few additional language constructs are discussed which provide functionality required by certain signs.

13.4.1 State transition customization

There are cases where it is necessary to generate advice based on the sign segmentation. For example, some signs such as “bath” (see Figure 13.3) contain a number of repetitions of the same action. In order to cater for such repetition, changes must be made to the allowable state transitions, starting states and stopping states. Figure 13.3 contains an example of the mechanism made available by the domain-specific language to specify such custom state transition behaviour.

To comment on the number of repetitions, a function must be defined which considers the sign segmentation and, for example, counts the number of repetitions of the action, generating advice if the number of repetitions is too small or too large. This is referred to as *segmentation level advice*. It is relatively independent of the single-frame advice generation system discussed earlier.

13.4.2 Feedback groups

Occasionally, it is helpful to treat a group of constraints as a single unit. For this purpose, the domain specific language provides the `feedback-group` construct. A `feedback-group` has the form

```
("feedback-group"
   annotations
   compound-constraint
   compound-constraint
   ...)
```

A `feedback-group` operates as an and-connective would, except that it labels all constraints nested within it using the provided annotations. Thus, specialized advice procedures may be applied to entire groups of constraints. For example, in the body

```
(palm-facing-upward hand)
(palm-pointing-left hand)
(feedback-group hand-shape
   (finger-flat (index hand))
   (finger-flat (middle hand))
   ...)
```

All the `finger-flat` constraints are labelled as being part of the hand shape. This was used during experimental work to automatically separate hand-shape advice from all other advice (the reason for this will be discussed in Section 14.3).

13.4.3 Named compound constraints

It is sometimes convenient to assign a name to a compound constraint to avoid unnecessary repetition in defining a sign. For example, the sign "woman" discussed earlier made use of the named constraint `hand-pose-woman` which is defined as follows

```
(defconstraint hand-pose-woman (hand)
  (palm-facing-upward hand)
  (palm-pointing-left hand)
  (feedback-group hand-shape
    (hand-flat hand)))
```

Named constraints can define a number of parameters. In this case, there is a single parameter `hand`. The sign "woman" passes `right-hand` as the parameter to the named constraint, which replaces all instances of `hand` within the body of the named constraint with `right-hand`. Note that the top-level constraints within the body of a named constraint are combined using and-semantics.

13.5 Summary

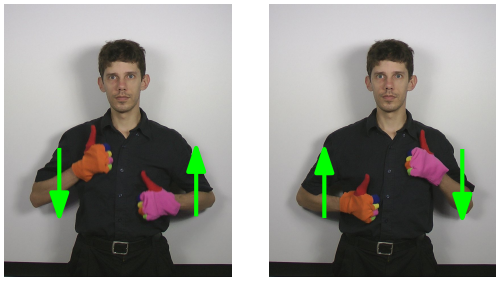
This chapter has considered feedback generation for whole video sequences. In order to achieve this, a domain-specific language that allows entire signs to be defined was described. This language includes:

- `interval` clauses for specifying the constraints particular to each phase of a sign;
- `start-when`/`stop-when` clauses for automatically detecting the meaningful portion of a sign;
- interval parameters and `transitions` clauses for customising state transition behaviour; and

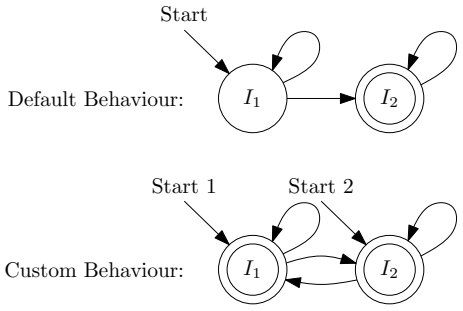
- `feedback-group` clauses that allow entire groups of constraints to be treated as a single unit.

It was demonstrated how this description could be used to segment a video sequence into phases using a form of Viterbi alignment, and how representative frames can be chosen from each of these phases. Subsequently, single-frame advice generation is applied to these representative frames in order to provide the user with feedback.

In the next chapter, the results obtained of an experimental evaluation of the feedback generation system using real video sequences are presented.



(a)



(b)

```
(defsign bath
  ...
  (interval ((label :first-pose)
             (stop-score 0))
            ...)
  (interval ((label :second-pose)
             (start-score 0))
            ...)
  ...
  (transitions
    (-1 :first-pose :second-pose)
    (-1 :second-pose :first-pose)))
```

(c)

Figure 13.3: (a) The SASL sign for "bath" is a two-handed sign which includes a number of repetitions of the same action. In (b), the HMMs corresponding to the default behaviour and the desired custom behaviour are shown. States are represented as circles, with allowable termination states being indicated by two concentric circles. Permitted state transitions are indicated using arrows between states. Possible starting states are indicated by inbound arrows with no source state. As shown in the figure, the default behaviour only allows starting in the first state I_1, and allows only one transition to I_2 which must also be the termination state. As such, the default sign segmentation behaviour is not desirable, because the signer must be able to start in either of the two states, and also to alternate between the states as needed. The definition in (c) demonstrates how to implement the required customisations using interval parameters and a transitions clause. Specifying a `start-score`/`stop-score` of 0 means that starting/terminating in the specific interval is allowable (using 0 specifically means there is no penalty for starting/stopping in the state). The transitions clause allows one to change the state transition pseudo-log-probabilities. Each interval must be referenced using the name specified with the `label` parameter passed to the interval. An entry such as `(-1 :second-pose :first-pose)` means that the transition pseudo-log-probability between the states labelled `:second-pose` and `:first-pose` should now be -1. This in turn means that the transition is allowed, but has a cost of 1.

Chapter 14

Experimental Evaluation

In the preceding chapters, the advice generation subsystem of the proof of concept system was presented. It was shown how compound constraints can be built from atomic constraints, and how compound constraints can be used to generate advisory comments based on the features obtained from a single frame of video. Using this process as a subcomponent, the definition of signs was discussed, and the means by which commentary may be generated based on entire video sequences was presented.

In order to demonstrate the efficacy of this system, a series of tests were performed using real video data collected from signers attempting six different South African Sign Language (SASL) signs.

14.1 Data collection

Video recordings were made using a Panasonic HDC-TM900 camera, which produced video with a resolution of 1920 by 1080 at 50 frames per second. Video was captured using a Blackmagic Design [14] Decklink HD Extreme capture card and stored in uncompressed YCbCr format using 10 bits per channel.[1]

Subjects wearing black clothing were recorded in a well-lit environment against a uniform background. Care was taken to ensure that motion blur was negligible by lowering camera exposure time while maintaining sufficient illumination to clearly distinguish glove markers.

Video data was collected from five different signers, each performing six different SASL signs. For each sign, each signer made a set of correct attempts. The signs are the words for “woman”, “sick”, “please”, “tall”, “now” and “bath”. Because the purpose of the study was to specifically demonstrate the functioning of the feedback generation system, each signer also made videos where they made a predefined set of errors.

Including all signers, the number of videos collected for “woman”, “sick”, “please”, “tall”, “now” and “bath” were 17, 29, 20, 30, 19 and 17 respectively, with 132 in total.

[1] Due to limitations of the capture card related to bus bandwidth, capturing uncompressed 1080p50 video in raw RGB format was not possible, hence the video has chroma subsampling.

14.2 Methods

For each of the signs mentioned, a domain specific language description was created. These descriptions are given in full in Section B.2. The signs were chosen to demonstrate the manner in which the tutor system applies a variety of constraints.

Hand pose estimation was performed on each of the videos. The tutor was applied to the pose estimates obtained from each of the recorded videos using the description of the corresponding sign to generate advice. Statistics were collected relating to the advice generated by the system in each case.

In cases where the user made no error, no advice was expected. If advice was generated by the tutor in such cases, the advice was recorded, and the user's attempt re-examined to note whether the advice targeted legitimate (though unrequested) errors in the user's motion. For erroneous signing, there was a set of expected advice stemming from the predefined errors made during signing. The advice generated was examined for these expected pieces of advice, and any omissions recorded.

In addition to expected output, there are also unexpected items of advice generated by the system. These can be categorized into two groups. The first group contains items of advice which, though unexpected, are in fact correct upon examination of the user's attempt. The second group of unexpected advice are those items which are incorrect.

It should be emphasised, when incorrect advice is generated, it is not a failure of the single-frame advice generation procedure, but rather due to inaccuracies in estimating input features, poor sign segmentation or inadequacies in the definition of either the relevant atomic constraints or the definition of the sign.

During the course of the experimental evaluation of the tutor, it was found that the finger joint angles based on the current hand pose estimation system implementation are still often too noisy to produce meaningful advice regarding the user's hand shape. For this reason the finger joint angles are used to demonstrate the function of the `feedback-group` construct. Each constraint on the finger joint angles was annotated in a similar fashion to the following

```
(defconstraint right-hand-pose-woman ()
  (palm-facing-upward right-hand)
  (palm-pointing-left right-hand)
  (feedback-group :hand-shape
     (finger-flat (index hand))
     (finger-flat (middle hand))
     (finger-flat (ring hand))
     (finger-flat (pinky hand))
     (thumb-bending-outward hand)
     (or (thumb-unit hand)
         (thumb-spread hand))))
```

In this way, finger joint angle constraints such as `finger-flat` were labelled and could be treated separately from other constraints such as `palm-facing-upward`. Advice was generated for such constraints, but these advice items were used to generate a single statistic, the percentage of shape violations. This number represents the average percentage of finger joint angle constraints violated. Because none of the errors considered include errors in the hand shape, this serves as an indicator of the performance of the pose estimation system's reconstruction of the hand shape.

Another reason for the inclusion of information related to hand shape is the importance of demonstrating that the system can, in principle, generate advice for these features. The finger joint angles add a large number of degrees of freedom to the feature space, and so it is critical to demonstrate that the system can effectively deal with the large increase in the feature space dimensionality. Ultimately, the system does produce correct advice for the joint angles *given that the input features are correctly estimated*. The issue is thus one of the input hand pose estimation system needing further refinement.

14.3 Results

In Tables 14.1 and 14.2, these results for the SASL signs "woman", "sick", "please", "tall", "now" and "bath" are presented. For each sign, the actions associated with the sign are demonstrated in the 'Actions' column.

In each case, the specific error or lack of one is listed under the 'Error' column. A frame of video from the erroneous attempt is shown in each case to demonstrate the nature of the error. The number of videos tested in each case is listed under the 'Num. Videos' column. Note that all five signers attempted each of the correct signs and erroneous signs, and so 'Num. Videos' combines all the attempts from every signer.

In each case, the advice the system is expected to produce based on the input sign and the specified error is listed under 'Expected Advice'. The success of the system in producing the expected pieces of advice is related in the 'Expected Present' column. Here the percentage of the expected advice which was correctly generated is shown. In the cases where the user was instructed to perform the sign correctly, these data items are not applicable, and so are the corresponding table entries are left empty.

The percentage of intervals unexpected, but correct, advice is listed under the 'Unexpected / Correct' column. The percentage of intervals containing incorrect, unexpected items of advice is given under the column 'Unexpected / Incorrect'. These percentages do not include errors based on hand shape, as explained in Section 14.2. Advice was generated for such constraints, but these advice items were used to generate a single statistic listed in the column '% Shape Violations'. This number represents the average percentage of finger joint angle constraints violated.

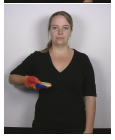
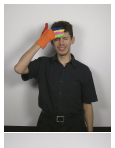
Sign	Actions	Error		Num. Videos	Advice				
					Expected Advice	% of Intervals			% Shape Violations
						Expected Present	Unexpected		
							Correct	Incorrect	
Woman		No error		9	—	—	0%	19%	14%
		Too far down		4	Your right hand must be at the same height as your right chest. Move your right hand upwards.	100%	17%	8%	15%
		Palm faces back		4	The palm of your right hand should be facing upwards. Rotate the palm of your right hand until it is facing upwards.	92%	0%	8%	20%
Sick		No error		15	—	—	0%	33%	20%
		Only forehead		5	Your right hand should be close to your stomach. Move your right hand to the *...(direction depends on last keyframe chosen)*	100%	27%	27%	40%
		Only stomach		4	Your right hand should be close to your head. Move your right hand upwards. Your right hand should be moving downward. Don't keep your right hand still, move it downwards.	100%	0%	41%	25%
		Palm faces down		5	The palm of your right hand should be facing backwards. Rotate the palm of your right hand until it is facing backwards.	100%	0%	33%	36%
Please		No error		15	—	—	13%	11%	30%
		No second hold		5	The downward motion of your hand should be followed by a pause, with the hand shaped like a fist. *Advice is generated on segmentation level. Note downward motion to neutral position still occurs.*	100%	10%	10%	25%

Table 14.1: Experimental results from tutor.

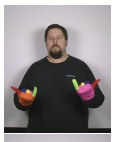
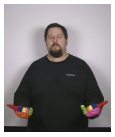
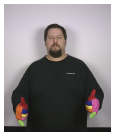
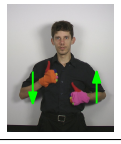
Sign	Actions	Error		Num. Videos	Advice				
					Expected Advice	% of Intervals			% Shape Violations
						Expected Present	Unexpected Correct	Unexpected Incorrect	
Tall	  	No error		15	—	—	2%	11%	17%
		Hand too far left		5	Your right hand must not be at the same horizontal position as your head. Move your right hand to the right.	100%	13%	0%	17%
		Palm facing forward		5	The palm of your right hand should be facing left. Rotate the palm of your right hand until it is facing left.	100%	0%	0%	14%
		Both previous errors		5	*Both previous items of expected advice.*	95%	0%	13%	14%
Now	  Repeated	No error		9	—	—	0%	15%	17%
		Palms facing inward		5	The palm of your right/left hand should be facing upwards. Rotate the palm of your right/left hand until it is facing upwards.	97%	20%	7%	15%
		Alternating		5	Your left hand should be moving downward/upward. Change the direction of movement of your left hand downwards/upwards.	80%	7%	33%	22%
Bath	  Repeated	No error		7	—	—	0%	42%	18%
		Non-alternating		5	Your left hand should be moving downward/upward. Change the direction of movement of your left hand downwards/upwards.	96%	0%	50%	22%
		No repetitions		5	Too few repetitions occurred. *Advice is generated on segmentation level.*	100%	0%	20%	20%

Table 14.2: Experimental results from tutor *(cont.)*

Type	Constraint tested	Sign	Error
Position	`(at-height left-chest right-hand)`	Woman	Right hand is too far down.
	`(close-to stomach right-hand)`	Sick	Hand touches forehead, but not stomach.
	`(close-to head right-hand)`	Sick	Hand touches stomach, but not forehead.
	`(not (same-x head right-hand))`	Tall	Hand too far left.
	`(righter-than head right-hand)`	Tall	Hand too far left.
Velocity	`(not (moving-horizontally right-hand))`	Sick	Hand touches forehead, but not stomach.
	`(moving-downward right-hand)`	Sick	Hand touches stomach, but not forehead.
	`(moving-upward left-hand)`	Now	Alternating movements.
	`(moving-downward left-hand)`	Now	Alternating movements.
	`(moving-upward left-hand)`	Bath	Non-alternating movements.
	`(moving-downward left-hand)`	Bath	Non-alternating movements.
Orientation	`(palm-facing-upward right-hand)`	Woman	Palm faces backward.
	`(palm-facing-backward right-hand)`	Sick	Palm faces downward.
	`(palm-facing-left right-hand)`	Tall	Palm faces forward.
	`(palm-facing-upward right-hand)`	Now	Palms facing inward.
	`(palm-facing-upward left-hand)`	Now	Palms facing inward.
Temporal	Implicit (default transitions)	Sick	Hand touches forehead, but not stomach.
	Implicit (default transitions)	Sick	Hand touches stomach, but not forehead.
	Segmentation level (default transitions)	Please	No second hold.
	Segmentation level (custom transitions)	Bath	No repetitions.

Table 14.3: All actively violated constraints tested during the experimental evaluation, classified according to type. Each constraint is listed with the sign and error that tests it. For clarity, it is noted when a constraint is directly inverted. In the case of temporal constraints, the constraint is classified according to whether the errors are detected implicitly during sign segmentation and commentary generation, or whether post-processing tests were performed at segmentation level. Use of customised state transition behaviour is indicated. Position, velocity and orientation constraints enforcing temporal constraints implicitly are listed with others of similar type.

14.4 Discussion

It is notable that the system was consistent in generating expected items of advice. The lowest percentage of expected advice being present was 80% for a single case in the sign "now", and was above 90% for all other cases.

Unexpected advice exhibited a higher degree of variability. Incorrect advice on hand orientation was somewhat more prevalent than other types of incorrect advice. This is attributed to hand orientation being the most noisy parameters after the hand joint angles. Position and velocity were well estimated, but occasional incorrect advice occurred in situations where the representative frame selected during sign segmentation favoured other constraints.

A fair proportion of the unexpected advice proved, in fact, to be correct. For example, the last frame in the incorrect attempt pictured in Figure 13.2b produced an additional comment regarding the orientation of the hand. It can be seen in the figure that the user's hand is not properly level, and so this advice was appropriate.

A variety of constraint types were tested. These included constraints on hand position, velocity and orientation. Temporal constraints are also tested, for example, by omitting phases of a sign. Temporal constraints were either enforced implicitly by the segmentation and commentary process, or explicitly by processing the final segmentation for high-level information such as repetition. Constraints that were intentionally violated are shown in Table 14.3 (the full definitions of the signs in Section B.2 may be consulted for other constraints used during

testing). The system's performance with respect to these different types of constraints will now be discussed.

14.4.1 Position Constraints

Position constraints tested included cases such as that in "woman" where the signer's hand was too far down. The constraint violated in this case was `(at-height left/right-chest right-hand)`. The system generated the expected advice in all cases. Note that the dynamic advice "Move your right hand upwards." indicates that the system knows that the reason the constraint was broken, was that the hand was too low. If the hand were at the height of the head, the dynamic advice would have been "Move your right hand downwards."

14.4.2 Velocity Constraints

Velocity constraints tested included relatively complex cases such as in "now" and "bath" where the velocities of both hands are constrained. It was found that one can prioritise the right hand by exploiting the substitution of the and-connectives with the avg-connectives. Below is a section adapted from the description of "bath":

```
(and
  (moving-downward right-hand)
  (and (moving-upward left-hand)
       (hands-bath)))
```

This is logically equivalent to joining all three constraints using a single and-connective. However, during segmentation the substitution of and-connectives with avg-connectives causes the constraint `(moving-downward right-hand)` to be weighted by 50%, and the inner constraints are each weighted 25%. This has the effect of biasing the segmentation towards triggering on the right hand. This is why the expected advice in the non-alternating movements case of "bath" is left hand specific. The segmentation procedure favours the right hand's constraint satisfaction, and so it is the constraint for the left hand that ultimately generates advice.

14.4.3 Orientation Constraints

Hand orientation constraints of various types were tested. One such case is reported for the sign "tall", where the palm is facing forward instead of to the left, violating the constraint `(palm-facing-left right-hand)`. The dynamic advice in this case was "Rotate the palm of your right hand until it is facing left." While correct, this textual advice does not fully express the information available to the advice generation system. Textual feedback does not naturally lend itself to this type of constraint. Ideally, the dynamic advice would be replaced with an animation showing the user's hand rotating from its current orientation to the orientation

contained in the suggested features. However, the most difficult portion of the advice generation has been successfully performed (obtaining suggested features and a set of constraints needing advice generation), and so enough information is available to pass to an animation subsystem.

14.4.4 Temporal Constraints

Temporal advice was also considered. In certain cases, the sign definition itself in conjunction with the segmentation subsystem is sufficient to detect errors such as missing sign phases. For example, in the sign "sick", two types of omission were tested. The first omits the touching of the stomach and the second omits the touching of the forehead. In the first case, the touching of the forehead is immediately followed by a move to the neutral position. The segmentation subsystem automatically assigns a frame of the hand during its downward motion to the interval which requires a touching of the stomach. This results in advice such as "Your right hand should be close to your stomach. Move your right hand to the left." In the case where the touching of the forehead is omitted, the intervals specifying proximity to the forehead and a downward motion are violated, resulting respectively in advice such as "Your right hand should be close to your head. Move you right hand upwards" and "Your right hand should be moving downward. Don't keep your right hand still, move it downwards." Note in the last case that the dynamic advice is generated with knowledge of both the current velocity and the suggested velocity.

Segmentation level advice was explored in two cases. The first was during the sign "please", where the second hold of the hand was omitted. The segmentation of the video revealed that the second interval was under-represented, indicating that it was missing. A function using a simple threshold can be used to test this, generating the advice "The downward motion of your hand should be followed by a pause, with the hand shaped like a fist." Another case using segmentation level advice considered a lack of repetitions in the sign for "bath". In this case, a function was written which checks the sign segmentation for the number of changes of hand direction. If this is too low, there were not enough repetitions and the advice "Too few repetitions occurred" is generated.

14.5 Summary

In this chapter, an experimental evaluation of the feedback generated by the tutor system was performed using a set of real video sequences of six South African Sign Language words. Each sign was defined using the domain specific language, and the videos then tested a variety of constraints present within the definitions. These constraints included positional, velocity, orientation and temporal constraints. In the following chapter, conclusions are presented regarding the system in terms of its theoretical development in earlier chapters, and the experimental evaluation performed in this chapter.

Chapter 15

Summary and Conclusion for the Tutor

In this chapter, as in Chapter 9, a summary of the stages of the research process, the questions investigated and the significance of the findings is presented. A more compact presentation of the dissertation's core findings and its original contributions is given in Chapter 16, which includes a section on future work.

A sign language tutor with vision-based hand pose tracking was demonstrated which is capable of providing context-sensitive advice based on a user's attempt at signing. Such truly context-sensitive feedback is lacking in existing tutor systems. In this respect the proposed system represents a major contribution towards advancing the state of the art.

At the beginning of the investigation, it was noted that existing sign language tutor systems are unable to provide detailed corrective feedback because the way in which these systems model signs have inappropriate levels of granularity. Systems based on standard sign language recognition systems (classification-based approaches) operate at the level of entire words, and so are unable to comment on the substructure of a word. Such systems can determine whether confusion has occurred between a set of possible words, or instead teach aspects of sentence-level sign language grammar, but are not able to address specific aspects of individual words.

In contrast, template based systems, which use dynamic time warping to fit an input sign to a pre-existing exemplar, have too fine a level of granularity. Each frame of the template represents a world state against which the user's attempt is aligned. However, at this fine level, the high-level phonological significance of the movements is lost, and so providing advice is limited to commenting on the proximity within state space of the input and the template.

This part of the dissertation was based on the observation that sign language words exhibit a phonological substructure, and that each phonological aspect can be commented on in a variety of ways such as natural language or diagrams. In particular, incorrect pronunciation can also be illustrated in these ways with reference to a word's phonology. It was noted that phonemes are characterised by a set of constraints on the signer, whether this is made explicit or not. The aim of the research was to determine whether meaningful commentary could be provided to the user of a sign language tutor based on a high-level description of the constraints imposed on a signer during the course of a sign, whether this commentary could be tailored to the specific

attempt made by the user, and whether the system could provide targeted advice with which the user can correct their mistake in a manner that would require the least possible effort.

It was realised that the constraints placed on a user during signing could be seen as an assignment of world states to scores quantifying the level of satisfaction an expert would assign to these states. Individual constraints relate to different, and often independent, aspects of the world state. This leads immediately to the idea of atomic constraints, which are functions mapping aspects of the world state to levels of satisfaction. Combinations of the atomic constraints can then be used to form more complex constraints on the signer.

Classification-based systems can be seen as using a single atomic constraint at word-level, meaning that the substructure of the sign is hidden, and so commentary on the intra-sign level is hampered. Template-based systems operate at the other extreme, where each frame of the template is a new atomic constraint. However, no high-level meaning is attached to these atomic constraints beyond that the user must match the template frame as well as possible. The system presented in this dissertation occupies the middle ground between these two extremes. At this level, meaningful constraints such as that "the right hand must be close to the right shoulder" may be specified. When the user has violated the constraint, corrective advice may be generated such as "Raise your right hand." if the user's hand was too low.

As multiple constraints are generally placed on the user's movements during a given phase of a sign, a means of combining atomic constraints was required. For this purpose, a domain specific language was defined which allows the combination of atomic constraints using a variety of logical connectives. While this form of specification is more flexible than any description framework used by existing sign language tutors, the flexibility led to key questions that were addressed during the course of the research. The most important of these is whether it is possible to use this linguistic description of a sign to generate commentary that is not merely descriptive of the error, but also provides targeted advice that can be easily understood by the user, is not self-contradictory, and can help the user correct their error.

The desired characteristics for targeted advice place stringent requirements on the advice generation system. While it is relatively simple to generate advice for single atomic constraints, the logical interaction of atomic constraints introduces complex dependencies that, initially, seem difficult to disentangle. Even if, for example, logically correct natural language sentences could be generated directly from this structure, this would not necessarily be in a form that the user can readily comprehend.

In response to this, the processes of not-propagation, or-pruning and and-prioritisation were developed which can reduce complex constraint structures into a list of simple constraints with possible inversion. This process, particularly the or-pruning stage, uses a set of features as a reference point for determining the active constraints for inclusion in the final list of constraints. As the list of constraints is interpreted using and-semantics, this is also a form readily accessible to a user, effectively a to-do list where all the constraints must be obeyed together. This represented a key contribution, because it enables the reduction of arbitrarily complex combinations

of constraints to this simple form.

Initially it appeared that the user's own attempt could be used directly during or-pruning for determining the active constraints for inclusion in the final list of constraints. However, upon further reflection, it was realised that this would be a possible source of paradoxical advice, and that the reference point must be a point where the combined constraint tree is satisfied. This point, the suggested features, represent the "goal" towards which advice attempts to steer the user. However, solving a pure constraint satisfaction problem to find the suggested features is insufficient for advice generation, as the resulting point may be distant in the state space, meaning the effort on the part of the user may be excessive. In response to this challenge, a measure of the user's effort was defined which could be embedded within the constraint satisfaction framework, allowing the constrained optimisation to be formulated as a series of constraint satisfaction problems. This allows constraints to be defined with less consideration for the subsequent constrained optimisation problem.

While the addition of the search for the suggested features introduced added complexity, the result was a more powerful advice generation system. Because the suggested features are known to be compatible with each of the constraints in the final list of constraints, each atomic constraint can generate advice that is to a large extent independent of the other constraints, as consensus has already been achieved. Therefore, advice generation can be defined for each constraint separately, without a need to define a means of combining them. This allows modes of feedback such as diagrams, which are not normally amenable to combination, to be used during advice generation.

The preceding process treats single frames of video, and so an additional subsystem was introduced to generate advice for entire video sequences. At its core, this process utilised the segmentation ability of hidden Markov models to segment the signs into intervals, and syntax was introduced into the domain-specific language to specify each interval's constraint tree, as well as customised transition behaviour.

The system was tested on 132 video sequences of correct and incorrect exemplars of 6 SASL signs. While this is a relatively small lexicon, the primary purpose of the experimental work was to investigate the advice generation capability of the system for a variety of constraint types. The experimental work established that the system is capable of generating the targeted feedback which motivated its creation. While hand shape data obtained from the pose estimation system is as yet too noisy for the generation of meaningful feedback, several other constraint types were tested, including position, velocity, orientation and temporal constraints.

It was observed that the advice generation subsystem successfully takes into account the current user attempt, and the least effort means of correcting their mistake. For example, if the right hand was required to be at chest height, but was in front of the stomach, advice was generated such as: "Your right hand should be at the same height as your chest. Move your right hand upwards". This demonstrated that system is aware about the current user attempt, and that the hand is too low. In addition, the system knows that the easiest way for the user to

correct their mistake is to move their right hand directly upwards.

In contrast, a classification-based system might give a low score for the word in its entirety, but not give an indication of what might be wrong. A template-based system might provide an animation synchronising the user's attempt to the template sequence, but would not be able to provide additional information regarding the degree of conformity required, and what aspects of movement are actually important, beyond simple distance thresholds within state space. While more useful than a simple global score, this feedback still lacks critical information required for a user to develop an understanding of the constraints on their movement.

This contrast highlights what gains have been made as the result of the research process. A system has been created that is able to reason about the constraints on the user's motions and the user's current attempt to dynamically generate corrective minimum-effort advice. These results are encouraging, and serve as ample motivation for further research. In the following chapter, the conclusion for the dissertation as a whole is presented, which will include additional reflection on the system as a whole, and a discussion of the potential future work arising from this investigation.

Part III

Conclusion and Future Work

Chapter 16

Overall Conclusion and Future Work

In this chapter, key questions considered within this dissertation are summarised, and the original contributions derived from addressing these questions are presented. The significance of these contributions is discussed, and possibilities for future work stemming from these conclusions are also discussed. This conclusion complements the individual conclusions for the pose estimation subsystem presented in Chapter 9, and the advice generation subsystem presented in Chapter 15, which reflected on these stages of the research.

16.1 Review of conducted work

16.1.1 Hand pose estimation

The first part of the dissertation focussed on the creation of a hand pose estimation subsystem which can serve as the frontend for the tutor subsystem. A tutor system places stringent requirements on the accuracy of hand pose estimates, which led to the selection of database driven pose estimation systems as the family of systems most appropriate to the task. The Earth Mover's Distance (EMD) was chosen as a similarity metric because of its ability to respond to detailed marker contour information. Two key questions arose from this selection. Firstly, whether the EMD would outperform a related system, establishing that its use is desirable. Secondly, whether the EMD could be effectively approximated by a two-tier search strategy where a rough metric is used to find a set of initial nearest neighbours, followed by a refinement step using the EMD, establishing that its use is computationally feasible.

The results presented in Chapter 6 answered both of these questions in the affirmative, where testing was performed on 500 real images with ground truth data. In particular, Figure 6.3 established that the compound EMD outperforms a system similar to that presented by Athitsos et al. [8][1]. This demonstrated that the EMD could outperform a related system, but an efficient approximation is needed to makes its use computationally feasible. The results presented in

[1]This earlier version of the systems presented by Athitsos et al. is referred to because later versions of the system represented fast *approximations* of this exact version with expanded databases. The same database was used to compare this system to the one based on the compound EMD.

Figure 6.4 showed that a two-tier search could, for modest numbers of initial nearest neighbours, closely approximate exact search using the compound EMD. While the approximate search is not yet fast enough for real-time use, a speed increase of two orders of magnitude was achieved.

A possible limitation on these results derives from the fact that a subset of the full tutor's pose estimation database viewpoints was used during the comparison. The direct use of the EMD and chamfer distance necessitated this for the purposes of the comparison, as otherwise the excessive query resolution time would have made the investigation impractical. This limitation is, however, partly mitigated by the fact that the most likely confusers of a given hand pose are those with similar viewpoint, and so these were present in the database.

By answering both key questions in the affirmative, the utility of the EMD as a dissimilarity metric in database driven hand pose estimation has been demonstrated, both in terms of its accuracy, and the ability to effectively approximate it.

It was noted that vision-based hand pose estimation systems using gloves often use colour model estimation systems which rely on ad hoc methods characterised by substantial manual intervention. As demonstrated in Chapter 7, automatic colour calibration can be performed by transforming rough prior knowledge regarding marker colour models and a calibration image of the hand in a known pose into refined posterior knowledge of the colour models that can lead to significant gains in marker segmentation performance, as demonstrated by the difference between Figures 7.17d and 7.17e.

It was noted that occasional segmentation problems such as missing regions still remain, and this could be attributable to the data scarcity in using a single calibration image. However, the colour calibration procedure substantially improved the quality of segmentation information available to the hand pose estimation system. It also serves as a more rigorous alternative to the ad hoc methods prevalent in existing glove-based hand pose estimation systems. Furthermore, graphical models are an inherently flexible framework, and so can easily serve as a foundation for further work on the colour calibration process.

16.1.2 Sign language tutor

The second part of the dissertation focussed on the challenge of generating corrective advice for the user of a sign language tutor. It was noted that existing tutor systems are not able to provide truly context-sensitive feedback to the user based on their current attempt.

In response, a framework was proposed where the constraints on a user's movement are modelled using atomic constraints which may be combined into complex logical structures. A domain specific language was defined which allows the combination of these atomic constraints using a variety of logical connectives. Furthermore, the language provides facilities for specifying multiple phases of a sign, and customise transitions between phases of the sign.

The significance of the domain specific language is twofold. Firstly, it allows the specification of aspects of a sign in detail. Secondly, it serves as a basis from which the system may

reason about a user's attempted sign, going beyond mere scoring of the attempt, and so allows the resulting system to surpass the abilities of existing systems.

A process was presented by which the system may reason about a user's input during a particular frame of video in the light of the linguistic description of the active phase of the sign. It was demonstrated that, despite the complexity introduced by the domain specific language's flexibility, the proposed processes of not-propagation, or-pruning and and-prioritisation can disentangle complex logical structures into a format that may be readily understood by the user. This is particularly important, as it transforms the domain specific language's flexibility from a potential burden into an asset.

As part of this simplification process, the system identifies a set of target actions (suggested features) as a goal towards which the user is encouraged by the generated advice. This aspect of the advice generation process is significant for two primary reasons. Firstly, the suggested features are selected so as to minimise the effort expended by the user in correcting their mistake. Secondly, the features are chosen so as to achieve consensus between separate constraints, and so avoiding potential sources of paradoxical advice. This means that advice may be generated for each constraint in a relatively independent fashion, meaning that no complex interactions between the advice generation mechanisms of each atomic constraint need to be taken into account, given the current features and the suggest features. This has important consequences for the flexibility of the system in terms of the exact means of output (for example, natural language, diagrams or animations), and its extensibility, as the addition of constraints and modes of advice do not require changes to existing advice generation mechanisms.

The capabilities of the system were demonstrated on a set of 132 video recordings of five signers signing six South African Sign Language signs. Despite the presence of some erroneous advice, expected advice was generated reliably, and demonstrated the context sensitive advice the system is capable of giving.

16.2 Future work and concluding remarks

Based on the results presented in the dissertation, a number of possible avenues for future research may be suggested.

- While experimental work has demonstrated that the tutor system can provide advice as required, usability studies still need to be performed in actual language learning environments. Curriculum development must be considered, and alternative modes of feedback, which would test the range of the sign definition system further.
- Monitoring of non-manual gestures including facial expression and body pose can be added to the tracker repertoire.
- Another similarity metric, which has a computational complexity and accuracy between the areas and means metric and the compound EMD, may be introduced within an addi-

tional layer to the search process. Mappings which can be run on a GPU would be particularly useful. A natural question that arises is whether one can distill the knowledge in the database and the similarity metric using a learning algorithm in such as metric. A possible avenue for future work would be to utilise a "Siamese" architecture [22] whereby a neural network is trained to embed features into a space where a simple metric such as L_1 can be used to make similar predictions to the original metric. The embedding due to Indyk and Thaper [38] could be used as a useful starting point for such a network, as it is effectively a linear mapping which can be adapted to the linear region of certain neural network architectures. This work would complement other work on approximating embeddings, which are an ongoing theme within the body of literature.

- The tracker in its current form is unable to deal with hands occluding each other. Interactions between the hands are quite common in sign language signs, so this aspect of the system needs to be addressed. The system is able to provide estimates of the hand poses before and after occlusions, so it would be possible to add a hand pose tracker that uses this information for initialisation purposes. The exact nature of this additional tracker would be the subject of investigation.

- As cameras which can provide depth data become more common, a natural question that arises is whether one can use similar techniques to those presented here on depth data. In principle, this would be relatively simple, as the EMD can be extended to an arbitrary number of dimensions. Possibly, this would allow unadorned hand tracking. However, this avenue of research would be highly dependent on having depth data of sufficient resolution that remains usable if the user is performing signs at speed, a subject that would require a feasibility study.

- While the tutor has demonstrated the ability to segment signs into phases given the sign's description, it was found that the Markov assumption underlying HMMs occasionally leads to "short-sightedness" when a sign is segmented. In effect, it is challenging to impose prior information about the *length* of each interval. Therefore, experimentation with other forms of stochastic grammar could be attempted. In particular, grammar production rules that are tree-based, rather than the effectively linear production rules imposed by HMMs, would be potentially fruitful.

- Atomic constraints in the tutor system are defined explicitly. It may be interesting to use these constraints as a type of prior, then employ data of recorded signing attempts to refine the constraints. The procedure used to generate a list of constraints for advice generation might be used to generate a list of constraints "responsible" for each moment in time, which could then be adjusted by the prior constraint models.

- Another question open for consideration is, given a set of constraints and a set of positive and negative video examples of a sign, whether a description of the the sign may be

obtained by data mining procedures. Such a procedure would have application in the existing tutor system, but may also prove useful in sign language phonology research.

In conclusion, the tutor is the first sign language tuition system to provide truly context-sensitive feedback to the user, significantly advancing the state of the art. It is also offers a variety of options for a future research program, whether this be in hand and body tracking, or in the e-teaching of sign language.

Appendices

Appendix A

Proof: The compound EMD is a metric

Equation 4.2.2 is restated below for convenience

$$d(h,h') = \sum_{k=1}^{6} d_{\text{EMD}}(s_k, s'_k).$$

We will briefly demonstrate that this dissimilarity measure is a metric by proving that, if $d_k(h,h')$ are metrics for $k \in \{1 \dots N\}$, then the sum of these metrics $d(h,h') = \sum_{k=1}^{N} d_k(h,h')$ is also a metric. In particular, we must show

$$\begin{aligned}
&d(h,h') \geq 0 \\
&d(h,h') = d(h',h) \\
&d(h,h') = 0 \iff h = h' \\
&d(h,h') \leq d(h,h^*) + d(h^*,h').
\end{aligned}$$

Firstly, $d(x,y) \geq 0$, because $d_k(h,h') \geq 0$ (d_k are metrics) and therefore their sum $d(x,y)$ must also be greater or equal to zero.

Because $d_k(h,h') = d_k(h',h)$, we have

$$\begin{aligned}
d(h,h') &= \sum_{k=1}^{N} d_k(h,h') \\
&= \sum_{k=1}^{N} d_k(h',h) \\
&= d(h',h)
\end{aligned}$$

Because $d_k(h,h') \geq 0$ and $d_k(h,h')$ is zero if and only if $h = h'$, the sum $d(h,h')$ is zero if and only if $h = h'$.

Lastly, the triangle inequality is shown using the fact that $d_k(h,h') \leq d_k(h,h^*) + d_k(h^*,h')$

$$\begin{aligned} d(h,h') &= \sum_{k=1}^{N} d_k(h,h') \\ &\leq \sum_{k=1}^{N} \left[d_k(h,h^*) + d_k(h^*,h') \right] \\ &= \sum_{k=1}^{N} d_k(h,h^*) + \sum_{k=1}^{N} d_k(h^*,h') \\ &= d(h,h^*) + d(h^*,h') \end{aligned}$$

If we set $N = 6$, identify $d_k(h,h') = d_{\text{EMD}}(s_k, s'_k)$, and observe that d_{EMD} is a metric, we conclude that d is a metric by the foregoing argument.

Appendix B

Signs and Constraints

B.1 List of constraints

A full list of constraints defined in the tutor subsystem is given in Table B.1.

The constraints `moving-upward` and `upward` differ in that `moving-upward` has a upward velocity threshold that needs to be exceeded, whereas `upward` is satisfied for any $v_y \geq 0$. So, `downward` can be defined simply as `(not (upward part))`, and so is a compound constraint. The constraint `moving-downward` cannot be defined in the same way, because inverting `moving-upward` would include a small region of positive velocities. For this reason, `moving-downward` is defined as an atomic constraint.

The constraints `touching`, `close-to` and `near` represent progressively slacker bounds on position. The constraint `far-from` is defined as `(not (near part other-part))`, and so is a compound constraint.

Type	Example Constraint	Definition	Variables Constrained
Position	`(higher-than part other-part)`	Atomic	p_y of both parts
	`(lower-than part other-part)`	Atomic	p_y of both parts
	`(lefter-than part other-part)`	Atomic	p_x of both parts
	`(righter-than part other-part)`	Atomic	p_x of both parts
	`(at-height part other-part)`	Atomic	p_y of both parts
	`(same-x part other-part)`	Atomic	p_x of both parts
	`(touching part other-part)`	Atomic	(p_x, p_y) of both parts
	`(close-to part other-part)`	Atomic	(p_x, p_y) of both parts
	`(near part other-part)`	Atomic	(p_x, p_y) of both parts
	`(far-from part other-part)`	Compound	(p_x, p_y) of both parts
Velocity	`(moving-upward part)`	Atomic	v_y of the part
	`(moving-downward part)`	Atomic	v_y of the part
	`(moving-leftward part)`	Atomic	v_x of the part
	`(moving-rightward part)`	Atomic	v_x of the part
	`(moving-horizontally part)`	Compound	v_x of the part
	`(moving-vertically part)`	Compound	v_y of the part
	`(moving part)`	Compound	(v_x, v_y) of the part
	`(unmoving part)`	Compound	(v_x, v_y) of the part
	`(upward part)`	Atomic	v_y of the part
	`(downward part)`	Compound	v_y of the part
	`(leftward part)`	Compound	v_x of the part
	`(rightward part)`	Atomic	v_x of the part
Orientation	`(palm-facing-left hand)`	Atomic	$\mathbf{o}_f$ of the hand
	`(palm-facing-right hand)`	Atomic	$\mathbf{o}_f$ of the hand
	`(palm-facing-upward hand)`	Atomic	$\mathbf{o}_f$ of the hand
	`(palm-facing-upward-right hand)`	Atomic	$\mathbf{o}_f$ of the hand
	`(palm-facing-upward-left hand)`	Atomic	$\mathbf{o}_f$ of the hand
	`(palm-facing-downward hand)`	Atomic	$\mathbf{o}_f$ of the hand
	`(palm-facing-downward-right hand)`	Atomic	$\mathbf{o}_f$ of the hand
	`(palm-facing-downward-left hand)`	Atomic	$\mathbf{o}_f$ of the hand
	`(palm-facing-forward hand)`	Atomic	$\mathbf{o}_f$ of the hand
	`(palm-facing-forward-left hand)`	Atomic	$\mathbf{o}_f$ of the hand
	`(palm-facing-forward-right hand)`	Atomic	$\mathbf{o}_f$ of the hand
	`(palm-facing-forward-upward hand)`	Atomic	$\mathbf{o}_f$ of the hand
	`(palm-facing-forward-upward-right hand)`	Atomic	$\mathbf{o}_f$ of the hand
	`(palm-facing-forward-upward-left hand)`	Atomic	$\mathbf{o}_f$ of the hand
	`(palm-facing-forward-downward hand)`	Atomic	$\mathbf{o}_f$ of the hand
	`(palm-facing-forward-downward-right hand)`	Atomic	$\mathbf{o}_f$ of the hand
	`(palm-facing-forward-downward-left hand)`	Atomic	$\mathbf{o}_f$ of the hand
	`(palm-facing-backward hand)`	Atomic	$\mathbf{o}_f$ of the hand
	`(palm-facing-backward-left hand)`	Atomic	$\mathbf{o}_f$ of the hand
	`(palm-facing-backward-right hand)`	Atomic	$\mathbf{o}_f$ of the hand
	`(palm-facing-backward-upward hand)`	Atomic	$\mathbf{o}_f$ of the hand
	`(palm-facing-backward-upward-right hand)`	Atomic	$\mathbf{o}_f$ of the hand
	`(palm-facing-backward-upward-left hand)`	Atomic	$\mathbf{o}_f$ of the hand
	`(palm-facing-backward-downward hand)`	Atomic	$\mathbf{o}_f$ of the hand
	`(palm-facing-backward-downward-right hand)`	Atomic	$\mathbf{o}_f$ of the hand
	`(palm-facing-backward-downward-left hand)`	Atomic	$\mathbf{o}_f$ of the hand

Table B.1: All constraints defined in the system, and the variables that they constrain. The terms `part` and `other-part` refer to named body parts (for example, the right hand, or the left shoulder). For the relevant body parts, p_x and p_y represent their positions, v_x and v_y their velocity, $\mathbf{o}_f$ and $\mathbf{o}_p$ the palm facing and pointing vectors (which determine the hand orientation), and j_n, the finger joint angles.

Type	Example Constraint	Definition	Variables Constrained
Orientation	`(palm-pointing-left hand)`	Atomic	$\mathbf{o}_p$ of the hand
	`(palm-pointing-right hand)`	Atomic	$\mathbf{o}_p$ of the hand
	`(palm-pointing-upward hand)`	Atomic	$\mathbf{o}_p$ of the hand
	`(palm-pointing-upward-right hand)`	Atomic	$\mathbf{o}_p$ of the hand
	`(palm-pointing-upward-left hand)`	Atomic	$\mathbf{o}_p$ of the hand
	`(palm-pointing-downward hand)`	Atomic	$\mathbf{o}_p$ of the hand
	`(palm-pointing-downward-right hand)`	Atomic	$\mathbf{o}_p$ of the hand
	`(palm-pointing-downward-left hand)`	Atomic	$\mathbf{o}_p$ of the hand
	`(palm-pointing-forward hand)`	Atomic	$\mathbf{o}_p$ of the hand
	`(palm-pointing-forward-left hand)`	Atomic	$\mathbf{o}_p$ of the hand
	`(palm-pointing-forward-right hand)`	Atomic	$\mathbf{o}_p$ of the hand
	`(palm-pointing-forward-upward hand)`	Atomic	$\mathbf{o}_p$ of the hand
	`(palm-pointing-forward-upward-right hand)`	Atomic	$\mathbf{o}_p$ of the hand
	`(palm-pointing-forward-upward-left hand)`	Atomic	$\mathbf{o}_p$ of the hand
	`(palm-pointing-forward-downward hand)`	Atomic	$\mathbf{o}_p$ of the hand
	`(palm-pointing-forward-downward-right hand)`	Atomic	$\mathbf{o}_p$ of the hand
	`(palm-pointing-forward-downward-left hand)`	Atomic	$\mathbf{o}_p$ of the hand
	`(palm-pointing-backward hand)`	Atomic	$\mathbf{o}_p$ of the hand
	`(palm-pointing-backward-left hand)`	Atomic	$\mathbf{o}_p$ of the hand
	`(palm-pointing-backward-right hand)`	Atomic	$\mathbf{o}_p$ of the hand
	`(palm-pointing-backward-upward hand)`	Atomic	$\mathbf{o}_p$ of the hand
	`(palm-pointing-backward-upward-right hand)`	Atomic	$\mathbf{o}_p$ of the hand
	`(palm-pointing-backward-upward-left hand)`	Atomic	$\mathbf{o}_p$ of the hand
	`(palm-pointing-backward-downward hand)`	Atomic	$\mathbf{o}_p$ of the hand
	`(palm-pointing-backward-downward-right hand)`	Atomic	$\mathbf{o}_p$ of the hand
	`(palm-pointing-backward-downward-left hand)`	Atomic	$\mathbf{o}_p$ of the hand
Hand Shape	`(thumb-unit finger)`	Atomic	j_0 of thumb
	`(thumb-spread finger)`	Atomic	j_0 of thumb
	`(thumb-out finger)`	Atomic	j_0 of thumb
	`(thumb-bending-outward finger)`	Atomic	j_1 of thumb
	`(thumb-bending-forward finger)`	Atomic	j_1 of thumb
	`(thumb-bending-across-palm finger)`	Atomic	j_1 of thumb
	`(first-knuckle-unbent finger)`	Atomic	j_1 of finger
	`(first-knuckle-bent-45-degrees finger)`	Atomic	j_1 of finger
	`(first-knuckle-bent-90-degrees finger)`	Atomic	j_1 of finger
	`(second-knuckle-unbent finger)`	Atomic	j_2 of finger
	`(second-knuckle-bent-45-degrees finger)`	Atomic	j_2 of finger
	`(second-knuckle-bent-90-degrees finger)`	Atomic	j_2 of finger
	`(finger-flat finger)`	Compound	All j_n of the finger
	`(finger-bent finger)`	Compound	All j_n of the finger
	`(finger-curled finger)`	Compound	All j_n of the finger
	`(hand-flat hand)` (used in woman, please and sick)	Compound	j_n of all fingers
	`(hand-fist hand)` (used in please)	Compound	j_n of all fingers
	`(hand-five-fingers-spread hand)`	Compound	j_n of all fingers
	`(hand-index hand)` (used in today)	Compound	j_n of all fingers
	`(hand-index-bent hand)`	Compound	j_n of all fingers
	`(hand-index-cup hand)`	Compound	j_n of all fingers
	`(hand-thumb hand)`	Compound	j_n of all fingers
	`(hand-index-middle-unit hand)`	Compound	j_n of all fingers
	`(hand-pinky-index hand)` (used in tall)	Compound	j_n of all fingers
	`(hand-baby-thumb hand)` (used in now)	Compound	j_n of all fingers
	`(hand-index-thumb-side hand)` (used in bath)	Compound	j_n of all fingers

Table B.1: All constraints defined in the system, and the variables that they constrain. The terms `part` and `other-part` refer to named body parts (for example, the right hand, or the left shoulder). For the relevant body parts, p_x and p_y represent their positions, v_x and v_y their velocity, $\mathbf{o}_f$ and $\mathbf{o}_p$ the palm facing and pointing vectors (which determine the hand orientation), and j_n, the finger joint angles.

B.2 Sign definitions

B.2.1 Woman

```
(defconstraint right-hand-pose-woman ()
  (palm-facing-upward right-hand)
  (palm-pointing-left right-hand)
  (feedback-group :hand-shape
     (hand-flat right-hand)))

(defsign woman
  (start-when
    (or (higher-than stomach right-hand)
        (at-height stomach right-hand))
    (unmoving right-hand))
  (interval ()
    (unmoving right-hand)
    (at-height left-chest right-hand)
    (same-x left-chest right-hand)
    (hand-pose-woman right-hand))
  (interval ()
    (not (moving-vertically right-hand))
    (moving-rightward right-hand)
    (at-height left-chest right-hand)
    (hand-pose-woman right-hand))
  (interval ()
    (unmoving right-hand)
    (at-height right-chest right-hand)
    (same-x right-chest right-hand)
    (hand-pose-woman right-hand))
  (stop-when
    (moving-downward right-hand)
    (lower-than stomach right-hand)))
```

B.2.2 Sick

```
(defconstraint hand-sick-high ()
  (or (palm-pointing-upward-left right-hand)
      (palm-pointing-left right-hand))
  (palm-facing-backward right-hand)
  (feedback-group :hand-shape
     (hand-flat right-hand)))

(defconstraint hand-sick-low ()
  (palm-pointing-left right-hand)
  (palm-facing-backward right-hand)
  (feedback-group :hand-shape
     (hand-flat right-hand)))

(defsign sick
    (start-when (unmoving right-hand)
                (or (at-height stomach right-hand)
                    (higher-than stomach right-hand)))
  (interval ()
    (unmoving right-hand)
    (hand-sick-high)
    (close-to head right-hand))
  (interval ()
    (higher-than stomach right-hand)
    (moving-downward right-hand)
    (and (not (moving-horizontally right-hand))
         (hand-sick-low)))
  (interval ()
    (unmoving right-hand)
    (close-to stomach right-hand)
    (hand-sick-low)))
```

B.2.3 Please

```
(defconstraint right-hand-pose-1-please ()
  (feedback-group :hand-shape
       (hand-flat right-hand))
  (or (palm-pointing-upward right-hand)
      (palm-pointing-upward-left right-hand))
  (or (palm-facing-backward right-hand)
      (palm-facing-backward-upward right-hand)))

(defconstraint right-hand-pose-2-please ()
  (feedback-group :hand-shape
       (hand-fist right-hand))
  (or (palm-pointing-forward right-hand)
      (palm-pointing-forward-upward right-hand))
  (or (palm-facing-upward right-hand)
      (palm-facing-backward-upward right-hand)))

(defsign please
  (start-when (close-to head right-hand))
  (interval ((label :please-first)
             (stop-score allow))
            (close-to head right-hand)
            (right-hand-pose-1-please))
  (interval ((stop-score allow))
            (not (moving-horizontally right-hand))
            (moving-downward right-hand)
            (or (right-hand-pose-1-please)
                (right-hand-pose-2-please)))
  (interval ()
            (unmoving right-hand)
            (not (higher-than left-chest right-hand))
            (right-hand-pose-2-please))
  (stop-when (and (not (at-height stomach right-hand))
                  (lower-than stomach right-hand))))
```

B.2.4 Tall

```
(defconstraint right-hand-pose-tall ()
  (feedback-group :hand-shape
                  (hand-pinky-index right-hand))
  (palm-pointing-upward right-hand)
  (palm-facing-left right-hand))

(defsign tall
  (start-when (higher-than stomach right-hand)
              (unmoving right-hand))
  (interval ()
            (or (at-height right-shoulder right-hand)
                (lower-than right-shoulder right-hand))
            (and (not (same-x head right-hand))
                 (righter-than head right-hand))
            (right-hand-pose-tall))
  (interval ()
            (not (moving-horizontally right-hand))
            (moving-upward right-hand)
            (right-hand-pose-tall))
  (interval ()
            (higher-than head right-hand)
            (unmoving right-hand)
            (right-hand-pose-tall))
  (stop-when (or (moving-downward right-hand)
                 (lower-than stomach right-hand))))
```

B.2.5 Now

```
(defconstraint hand-now (hand)
  (feedback-group :hand-shape
                  (hand-baby-thumb hand))
  (palm-facing-upward hand)
  (palm-pointing-forward hand))

(defconstraint hands-now ()
  (hand-now right-hand)
  (hand-now left-hand))

(defsign now
    (start-when (or (at-height stomach right-hand)
                    (higher-than stomach right-hand))
                (unmoving right-hand))
  (interval ()
            (at-height right-hand left-hand)
            (moving-downward right-hand)
            (and (moving-downward left-hand)
                 (hands-now)))
  (interval ()
            (at-height right-hand left-hand)
            (moving-upward right-hand)
            (and (moving-upward left-hand)
                 (hands-now)))
  (interval ()
            (at-height right-hand left-hand)
            (moving-downward right-hand)
            (and (moving-downward left-hand)
                 (hands-now)))
  (stop-when (unmoving right-hand)
             (unmoving left-hand)))
```

B.2.6 Bath

```
(defconstraint hand-bath (hand)
  (feedback-group :hand-shape
                  (hand-thumb hand))
  (or (palm-facing-backward hand)
      (palm-facing-backward-upward hand)
      (palm-facing-backward-downward hand)))

(defconstraint hands-bath ()
  (hand-bath right-hand)
  (or (palm-pointing-left right-hand)
      (palm-pointing-downward-left right-hand)
      (palm-pointing-upward-left right-hand))
  (hand-bath left-hand)
  (or (palm-pointing-right left-hand)
      (palm-pointing-downward-right left-hand)
      (palm-pointing-upward-right left-hand)))

(defsign bath
    (start-when (moving-downward right-hand)
                (higher-than stomach right-hand))
  (interval ((label :first-pose)
             (stop-score 0))
            (moving-downward right-hand)
            (and (moving-upward left-hand)
                 (hands-bath)))
  (interval ((label :second-pose)
             (start-score 0))
            (moving-upward right-hand)
            (and (moving-downward left-hand)
                 (hands-bath)))
  (stop-when (lower-than stomach left-hand)
             (lower-than stomach right-hand))
  (transitions
    (-1 :first-pose :second-pose)
    (-1 :second-pose :first-pose)))
```

List of References

[1] Achmed, I. and Connan, J. (2010). Upper body pose estimation towards the translation of South African Sign Language. In *Proceedings of the South African Telecommunication Networks and Applications Conference*. `http://www.satnac.org.za/proceedings/2010/papers/standardisation/AchmedFP381.pdf`.

[2] Akmeliawati, R., Ooi, M. P., and Kuang, Y. C. (2007). Real-time Malaysian Sign Language translation using colour segmentation and neural network. In *Proceedings of the Instrumentation and Measurement Conference*, pages 1169–1174. doi:10.1109/IMTC.2007.379311.

[3] Al-Jarrah, O. and Halawani, A. (2001). Recognition of gestures in Arabic Sign Language using neuro-fuzzy systems. *Artificial Intelligence*, 133(1-2):117–138, doi:10.1016/S0004-3702(01)00141-2.

[4] Aran, O., Ari, I., Akarun, L., Sankur, B., Benoit, A., Caplier, A., Campr, P., Carrillo, A. H., and Fanard, F. X. (2009). SignTutor: An interactive system for sign language tutoring. *IEEE Multimedia*, 16(1):81–93, doi:10.1109/MMUL.2009.17.

[5] Aran, O., Keskin, C., and Akarun, L. (2005). Sign language tutoring tool. In *Proceedings of the European Signal Processing Conference*, pages 2525–2528, Antalya, Turkey. `http://www.idiap.ch/~oaran/publications/aran05sign.pdf`.

[6] Athitsos, V., Alon, J., Sclaroff, S., and Kollios, G. (2004). Boostmap: A method for efficient approximate similarity rankings. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages II:268–275. IEEE Computer Society, doi:10.1109/CVPR.2004.52.

[7] Athitsos, V., Alon, J., Sclaroff, S., and Kollios, G. (2008). BoostMap: an embedding method for efficient nearest neighbor retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(1):89–104, doi:10.1109/TPAMI.2007.1140.

[8] Athitsos, V. and Sclaroff, S. (2003). Estimating 3D hand pose from a cluttered image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 432–439. IEEE Comput. Soc, doi:10.1109/CVPR.2003.1211500.

[9] Athitsos, V. and Sclaroff, S. (2004). Database indexing methods for 3D hand pose estimation. In Camurri, A. and Volpe, G., editors, *Gesture-Based Communication in Human-Computer Interaction*, pages 288–299. Springer, doi:10.1007/978-3-540-24598-8_27.

[10] Athitsos, V., Wang, H., and Stefan, A. (2010). A database-based framework for gesture recognition. *Personal and Ubiquitous Computing*, 14(6):511–526, doi:10.1007/s00779-009-0276-x.

[11] Barber, D. (2012). *Bayesian reasoning and machine learning*. Cambridge University Press.

[12] Bellegarda, J. and Nahamoo, D. (1989). Tied mixture continuous parameter models for large vocabulary isolated speech recognition. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 13–16. IEEE, doi:10.1109/ICASSP.1989.266351.

[13] Bishop, C. (2006). *Pattern recognition and machine learning*. Springer, New York.

[14] Blackmagic Design. Blackmagic Design Homepage. `http://www.blackmagicdesign.com`.

[15] Bowden, R., Windridge, D., Kadir, T., Zisserman, A., and Brady, M. (2004). A linguistic feature vector for the visual interpretation of sign language. In *Computer Vision-ECCV 2004*, pages 390–401. Springer, doi:10.1007/978-3-540-24670-1_30.

[16] Bradski, G. (1998). Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, Q2, `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.14.7673&rep=rep1&type=pdf`.

[17] Bradski, G. (2000). The OpenCV library. *Dr. Dobb's Journal of Software Tools*, 25(11):120–125.

[18] Brand, M., Oliver, N., and Pentland, A. (1997). Coupled hidden Markov models for complex action recognition. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 994–999. IEEE Comput. Soc, doi:10.1109/CVPR.1997.609450.

[19] Bungeroth, J., Stein, D., Dreuw, P., Ney, H., Morrissey, S., Way, A., and van Zijl, L. (2008). The ATIS sign language corpus. In *Proceedings of the International Conference on Language Resources and Evaluation*, pages 2943–2946. `http://www.lrec-conf.org/proceedings/lrec2008/pdf/748_paper.pdf`.

[20] Burger, T., Caplier, A., and Mancini, S. (2005). Cued Speech hand gestures recognition tool. In *Proceedings of the European Signal Processing Conference*. `http://www.eurasip.org/Proceedings/Eusipco/Eusipco2005/defevent/papers/cr2058.pdf`.

[21] Canny, J. (1986). A Computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, doi:10.1109/TPAMI.1986.4767851.

[22] Chopra, S., Hadsell, R., and LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 1:539–546, doi:10.1109/CVPR.2005.202.

[23] Curious Labs (2002). *Poser 5 Reference Manual*. Santa Cruz, CA, USA.

[24] de Villiers, H., van Zijl, L., and Niesler, T. (2012). Vision-based hand pose estimation through similarity search using the Earth Mover's Distance. *IET Computer Vision*, 6(4):285–295, doi:10.1049/iet-cvi.2011.0128.

[25] de Villiers, H. A. C., Niesler, T. R., and van Zijl, L. (2010). Balancing Runtime Space- and Time Complexity in Synthetic Database Driven Hand Posture Reconstruction Systems. In Nicolls, F., editor, *Proceedings of the Twenty-First Annual Symposium of the Pattern Recognition Association of South Africa*, pages 57–62, Stellenbosch, South Africa. `http://www.prasa.org/proceedings/2010/prasa2010-10.pdf`.

[26] de Villiers, H. A. C., van Zijl, L., and Niesler, T. R. (2013). Context-sensitive feedback for a vision-based sign language tutor. *Universal Access in the Information Society*, Submitted.

[27] Dick, T., Zieren, J., and Kraiss, K. (2006). Visual hand posture recognition in monocular image sequences. In *Proceedings of the German Association for Pattern Recognition Symposium*, pages 566–575, Berlin. Springer, doi:10.1007/11861898_57.

[28] Ellis, D., Singh, R., and Sivadas, S. (2001). Tandem acoustic modeling in large-vocabulary recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 517–520. IEEE, doi:10.1109/ICASSP.2001.940881.

[29] Erol, A., Bebis, G., Nicolescu, M., Boyle, R. D., and Twombly, X. (2007). Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding*, 108(1-2):52–73, doi:10.1016/j.cviu.2006.10.012.

[30] Fang, G., Gao, W., and Zhao, D. (2007). Large-vocabulary continuous sign language recognition based on transition-movement models. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 37(1):1–9, doi:10.1109/TSMCA.2006.886347.

[31] Fels, S. S. and Hinton, G. E. (1993). Glove-Talk: a neural network interface between a data-glove and a speech synthesizer. *IEEE Transactions on Neural Networks*, 4(1):2–8, doi:10.1109/72.182690.

[32] Freund, Y. and Schapire, R. (1995). A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, volume 139, pages 23–37. Springer, doi:10.1007/3-540-59119-2_166.

[33] Gao, W., Fang, G., Zhao, D., and Chen, Y. (2004). Transition movement models for large vocabulary continuous sign language recognition. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*, pages 553–558. Ieee, doi:10.1109/AFGR.2004.1301591.

[34] Grauman, K. and Darrell, T. (2004). Fast contour matching using approximate Earth Mover's Distance. In *Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 220–227, Washington, DC, USA. doi:10.1109/CVPR.2004.1315035.

[35] Grobel, K. and Assan, M. (1997). Isolated sign language recognition using hidden Markov models. *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, pages 162–167, doi:10.1109/ICSMC.1997.625742.

[36] Holden, E.-J., Lee, G., and Owens, R. (2005). Australian sign language recognition. *Machine Vision and Applications*, 16(5):312–320, doi:10.1007/s00138-005-0003-1.

[37] Huang, C.-L. and Huang, W.-Y. (1998). Sign language recognition using model-based tracking and a 3D Hopfield neural network. *Machine Vision and Applications*, 10(5-6):292–307, doi:10.1007/s001380050080.

[38] Indyk, P. and Thaper, N. (2003). Fast image retrieval via embeddings. In *Proceedings of the International Workshop on Statistical and Computational Theories of Vision (ICCV)*, Nice, France. `http://people.csail.mit.edu/indyk/emd.pdf`.

[39] Karpouzis, K., Caridakis, G., Fotinea, S.-E., and Efthimiou, E. (2007). Educational resources and implementation of a Greek sign language synthesis architecture. *Computers & Education*, 49(1):54–74, doi:10.1016/j.compedu.2005.06.004.

[40] Khronos Group. OpenCL Standard. `http://www.khronos.org/opencl/`.

[41] Kim, J. S., Jang, W., and Bien, Z. (1996). A dynamic gesture recognition system for the Korean sign language (KSL). *IEEE Transactions on Systems, Man, and Cybernetics. Part B: Cybernetics*, 26(2):354–359, doi:10.1109/3477.485888.

[42] Kim, T., Livescu, K., and Shakhnarovich, G. (2012). American sign language fingerspelling recognition with phonological feature-based tandem models. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 119–124. IEEE, doi:10.1109/SLT.2012.6424208.

[43] Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: Principles and techniques*. MIT Press.

[44] Kong, W. W. and Ranganath, S. (2008). Sign language phoneme transcription with rule-based hand trajectory segmentation. *Journal of Signal Processing Systems*, 59(2):211–222, doi:10.1007/s11265-008-0292-5.

[45] Liddell, S. K. and Johnson, R. E. (1989). American Sign Language: The phonological base. *Sign Language Studies*, 64:195–278.

[46] Lowe, D. (1999). Object recognition from local scale-invariant features. *Proceedings of the IEEE International Conference on Computer Vision*, 2:1150–1157, doi:10.1109/ICCV.1999.790410.

[47] Marshall, C. R. (2011). Sign Language Phonology. In Kula, N. C., Botma, B., and Nasukawa, K., editors, *The Continuum Companion to Phonology*, pages 254–277. Continuum International Publishing Group.

[48] Murakami, K. and Taguchi, H. (1991). Gesture recognition using recurrent neural networks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 237–242, New York, New York, USA. ACM Press, doi:10.1145/108844.108900.

[49] Naidoo, N. and Connan, J. (2009). Gesture recognition using feature vectors. In *Proceedings of the South African Telecommunication Networks and Applications Conference*, number 27. `http://satnac.org.za/proceedings/2009/papers/software/Paper58.pdf`.

[50] Nolker, C. and Ritter, H. (2002). Visual recognition of continuous hand postures. *IEEE Transactions on Neural Networks*, 13(4):983–94, doi:10.1109/TNN.2002.1021898.

[51] Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, doi:10.1109/5.18626.

[52] Rajah, C. (2006). Chereme-based recognition of isolated, dynamic gestures from South African Sign Language with hidden Markov models. `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.108.4576&rep=rep1&type=pdf`.

[53] Ren, Z., Yuan, J., and Zhang, Z. (2011). Robust hand gesture recognition based on Finger-Earth Mover's Distance with a commodity depth camera. In *Proceedings of the ACM International Conference on Multimedia*, page 1093, New York, New York, USA. ACM Press, doi:10.1145/2072298.2071946.

[54] Rigoll, G., Kosmala, A., and Eickeler, S. (1998). High performance real-time gesture recognition using Hidden Markov Models. In *Gesture and Sign Language in Human-Computer Interaction*, pages 69–80. Springer, doi:10.1007/BFb0052990.

[55] Rosales, R., Athitsos, V., Sigal, L., and Sclaroff, S. (2001). 3D hand pose reconstruction using specialized mappings. In *Proceedings IEEE International Conference on Computer Vision*, volume 1, pages 378–385. Boston University, doi:10.1109/ICCV.2001.937543.

[56] Rubner, Y., Tomasi, C., and Guibas, L. (1998). A metric for distributions with applications to image databases. In *Proceedings of the International Conference on Computer Vision*, pages 59–66, Bombay, India. doi:10.1109/ICCV.1998.710701.

[57] Rubner, Y., Tomasi, C., and Guibas, L. (2000). The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, doi:10.1023/A:1026543900054.

[58] Sagawa, H. and Takeuchi, M. (2002). A teaching system of Japanese sign language using sign language recognition and generation. In *Proceedings of the ACM International Conference on Multimedia*, pages 137–145, New York, New York, USA. ACM Press, doi:10.1145/641034.641035.

[59] Schröder, M., Elbrechter, C., Maycock, J., Haschke, R., Botsch, M., and Ritter, H. (2012). Real-Time Hand Tracking with a Color Glove for the Actuation of Anthropomorphic Robot Hands. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots*. `http://www.citmed.uni-bielefeld.de/publications/humanoids12.pdf`.

[60] Segno Communication CC (2013). SignGenius. `http://signgenius.com`.

[61] Siskind, J. M. and McAllester, D. A. (1993). Nondeterministic Lisp as a substrate for constraint logic programming. In *Proceedings of the National Conference on Artificial Intelligence*, pages 133–138. `ftp://www.ai.mit.edu/pub/screamer/aaai93.ps`.

[62] Slaney, M. and Casey, M. (2008). Locality-sensitive hashing for finding nearest neighbors. *IEEE Signal Processing Magazine*, 25(2):128–131, doi:10.1109/MSP.2007.914237.

[63] Sloane, N. J. A., Hardin, R. H., and Smith, W. D. Tables of spherical codes.

[64] Starner, T. and Pentland, A. (1995). Real-time American Sign Language recognition from video using hidden Markov models. In *Proceedings of International Symposium on Computer Vision*, pages 265–270. doi:10.1109/ISCV.1995.477012.

[65] Starner, T., Weaver, J., and Pentland, A. (1998). Real-time American sign language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1371–1375, doi:10.1109/34.735811.

[66] Stenger, B., Thayananthan, A., and Cipolla, R. (2004). Hand pose estimation using hierarchical detection. In *Proceedings of the International Workshop on Human Computer Interfaces*, pages 105–116, Prague, Czech Republic. doi:10.1007/978-3-540-24837-8_11.

[67] Teh, C.-H. and Chin, R. (1989). On the detection of dominant points on digital curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(8):859–872, doi:10.1109/34.31447.

[68] Torralba, A., Fergus, R., and Weiss, Y. (2008). Small codes and large image databases for recognition. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, doi:10.1109/CVPR.2008.4587633.

[69] van Zijl, L. (2006). South African Sign Language machine translation project. In *Proceedings of the International ACM SIGACCESS Conference on Computers and Accessibility*, page 233, New York, New York, USA. ACM Press, doi:10.1145/1168987.1169031.

[70] van Zijl, L. and Combrink, A. (2006). The South African Sign Language machine translation project: Issues on non-manual sign generation. In *Proceedings of SAICSIT06*, Somerset West, South Africa. doi:10.1145/1216262.1216276.

[71] van Zijl, L. and Olivrin, G. (2008). South African Sign Language assistive translation. In *Proceedings of the IASTED International Conference on Assistive Technologies*, Baltimore, USA. `http://www.cs.sun.ac.za/~lvzijl/publications/AT2008.pdf`.

[72] Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–511–I–518. doi:10.1109/CVPR.2001.990517.

[73] Vogler, C. and Metaxas, D. (1997). Adapting hidden Markov models for ASL recognition by using three-dimensional computer vision methods. *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, pages 156–161, doi:10.1109/ICSMC.1997.625741.

[74] Vogler, C. and Metaxas, D. (1998). ASL recognition based on a coupling between HMMs and 3D motion analysis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 363–369. doi:10.1109/ICCV.1998.710744.

[75] Vogler, C. and Metaxas, D. (1999a). Parallel hidden Markov models for American sign language recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1:116–122. doi:10.1109/ICCV.1999.791206.

[76] Vogler, C. and Metaxas, D. (1999b). Toward scalability in ASL recognition: Breaking down signs into phonemes. *Gesture-Based Communication in Human-Computer Interaction*, pages 211–224, doi:10.1007/3-540-46616-9_19.

[77] Vogler, C. and Metaxas, D. (2001). A Framework for Recognizing the Simultaneous Aspects of American Sign Language. *Computer Vision and Image Understanding*, 81(3):358–384, doi:10.1006/cviu.2000.0895.

[78] von Hardenberg, C. and Bérard, F. (2001). Bare-hand human-computer interaction. In *Proceedings of the Workshop on Perceptive User Interfaces*, New York, New York, USA. ACM Press, doi:10.1145/971478.971513.

[79] Walter, J. and Ritter, H. (1996). Rapid learning with parametrized self-organizing maps. *Neurocomputing*, 12(2-3):131–153, doi:10.1016/0925-2312(95)00117-4.

[80] Wang, R. Y. and Popović, J. (2009). Real-time hand-tracking with a color glove. *ACM Transactions on Graphics*, 28(3):1, doi:10.1145/1531326.1531369.

[81] Weber, R., Schek, H., and Blott, S. (1998). A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the International Conference on Very Large Data Bases*, pages 194–205. `http://www.vldb.org/conf/1998/p194.pdf`.

[82] Xu, L., Varadharajan, V., Maravich, J., Tongia, R., and Mostow, J. (2007). DeSIGN: An intelligent tutor to teach American Sign Language. In *Speech and Language Technology in Education*, pages 45–48. `http://www.ri.cmu.edu/pub_files/pub4/xu_ling_2007_1/xu_ling_2007_1.pdf`.

[83] Yamato, J., Ohya, J., and Ishii, K. (1992). Recognizing human action in time-sequential images using hidden Markov model. In *Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 379–385. IEEE Comput. Soc. Press, doi:10.1109/CVPR.1992.223161.

[84] Yang, R. and Sarkar, S. (2009). Coupled grouping and matching for sign and gesture recognition. *Computer Vision and Image Understanding*, 113(6):663–681, doi:10.1016/j.cviu.2008.09.005.

[85] Yianilos, P. (1993). Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 311–321, Austin, Texas, USA. `http://dl.acm.org/citation.cfm?id=313559.313789`.

[86] Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8(3):338–353, doi:10.1016/S0019-9958(65)90241-X.

[87] Zafrulla, Z., Brashear, H., Yin, P., Presti, P., Starner, T., and Hamilton, H. (2010). American sign language phrase verification in an educational game for Deaf children. In *Proceedings of the International Conference on Pattern Recognition*, pages 3846–3849. doi:10.1109/ICPR.2010.937.

[88] Zaki, M. M. and Shaheen, S. I. (2011). Sign language recognition using a combination of new vision based features. *Pattern Recognition Letters*, 32(4):572–577, doi:10.1016/j.patrec.2010.11.013.

[89] Zezula, P., Amato, G., Dohnal, V., and Batko, M. (2006). *Similarity search: The metric space approach*. Springer.

[90] Zhang, L.-G., Chen, Y., Fang, G., Chen, X., and Gao, W. (2004). A vision-based sign language recognition system using tied-mixture density HMM. In *Proceedings of the International Conference on Multimodal Interfaces*, pages 198–204, New York, New York, USA. ACM Press, doi:10.1145/1027933.1027967.

[91] Zieren, J. (2007). *Visuelle Erkennung von Handposituren für einen interaktiven Gebärdensprachtutor*. PhD thesis, RWTH Aachen. `http://darwin.bth.rwth-aachen.de/opus3/volltexte/2007/1975/`.

[92] Zimmerman, T. G., Lanier, J., Blanchard, C., Bryson, S., and Harvill, Y. (1987). A hand gesture interface device. In *Proceedings of the SIGCHI/GI Conference on Human Factors in Computing Systems and Graphics Interface*, pages 189–192, New York, New York, USA. ACM Press, doi:10.1145/29933.275628.